

BEYOND DATA:  
EFFICIENT KNOWLEDGE-GUIDED LEARNING  
FOR SPARSE AND STRUCTURED DOMAINS

by

Harsha Kokel

APPROVED BY SUPERVISORY COMMITTEE:

---

Sriraam Natarajan, Chair

---

Prasad Tadepalli

---

Vibhav Gogate


---

Rishabh Iyer

Copyright © 2023

Harsha Kokel

All rights reserved

*To my constant, steadfast,  
unwavering, loyal, and limitless supporter,  
Divya Kokel,  
Mummy .*

BEYOND DATA:  
EFFICIENT KNOWLEDGE-GUIDED LEARNING  
FOR SPARSE AND STRUCTURED DOMAINS

by

HARSHA KOKEL, BTech, MS

DISSERTATION

Presented to the Faculty of  
The University of Texas at Dallas  
in Partial Fulfillment  
of the Requirements  
for the Degree of

DOCTOR OF PHILOSOPHY IN  
COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2023

## ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to my advisor, Professor Sriraam Natarajan. His principles, work ethic, scholarship, hard work, academic service, and dedication have been a guiding light throughout my PhD. He leads by example and has inspired me to be my best self. His kind words have given me confidence when I doubted myself, his staunch support has protected me from the embarrassment of being dumbfounded after various presentations, his outlook on life and society has reminded me to be more patient and kind, his critiques have helped me grow, and his research doctrine has shaped my work. I am very fortunate that he moved to UT Dallas in 2017, the same year I started my master’s program. In all honesty, when I started working with Prof. Natarajan I used to have a really hard time understanding what he said. For more than a year, I struggled to see how *concept-learning* and *planning* were equivalent. His statements, “this is *almost exactly* the same as what we did  $X$  years ago,” baffled me countless times. However, now looking back at it, I see that these statements and observations helped me develop an important skill of thinking abstractly. To quote Ben Orlin, “To do good work, you’ve first got to engage with nitty-gritty details. Then, to do great work, you’ve got to move beyond them.” Prof. Natarajan’s mentorship and guidance have trained me to move beyond the nitty-gritty and honed my scholarly skills. Thank you, Prof. Natarajan, for expanding your patience each time I tested your limits.

I thank my supervisory committee, Prof. Prasad Tadepalli, Prof. Vibhav Gogate, and Prof. Rishabh Iyer. Prof. Tadepalli guided us in rewriting the initial draft of the RePReL paper. His direct yet polite questions helped organize the contributions of our work, and his attention to detail helped us tie loose ends. I learned the importance of keeping the paper crisp (without fanfare) and contributions precise. Prof. Vibhav Gogate is an excellent teacher. I took his machine learning course in 2018 and that got me through the qualifiers. He also graciously allowed me to sit through his PGM course, which has helped me in my

research. Finally, I have had the opportunity to co-author a paper with Prof. Rishabh Iyer which benefitted me in various ways.

I thank all my collaborators throughout this journey: Prof. Ravindran Balaraman, for the opportunity to learn the nuts and bolts of Reinforcement Learning; Dr. Shirin Sohrabi, for mentoring and championing me during my IBM internships; Dr. Michael Katz, for engaging me in the research discussions for hours and supporting my intuitions; and Dr. Kavitha Srinivas, for building my confidence in my research abilities.

Past and present members of the Starling lab have helped me throughout my PhD journey, professionally as well as personally. I express my gratitude to Alexander Hayes, Athresh Karanam, Dr. Brian Ricks, Dr. Devendra Dhami, Dr. Gautam Kunapuli, Kaushik Roy, Dr. Mayukh Das, Michael (Mike) Skinner M.D., Dr. Nandini Ramanan, Dr. Navdeep Kaur, Nikhilesh Prabhakar, Dr. Phillip Odom, Ranveer Singh, Sahil Sidheekh, Saurabh Mathur, Siwen Yan, Dr. Srijita Das, Dr. Shuo Yang, Dr. Yuqiao Chen, and Yuxin Zi. I have enjoyed my time and discussions with all of them. A special shout out to Mike who never failed to proofread my manuscripts and provided valuable timely feedback.

A special acknowledgment to Prof. Prasenjit Majumder at DAIICT. Born in a Sindhi business family, my sole intention when I first met him, in the summer of 2012, was to get a campus job that earned me a stipend to manage my expenses. Had it not been for him, I would have never known the joy of working on a research problem. He introduced me to the research community, funded my trip to Kolkata to attend FIRE in December 2012, and provided me with an opportunity to conduct a hands-on tutorial at Microsoft Research Bangalore in 2013. This research experience, his talks, and the time spent with him and the IR lab members made a deep impression on my young mind. He is the one who recruited me to the “cult of PhD”.

Many times during these last couple of years, especially during the pandemic, I could not conjure even a single reason to continue with the PhD. In fact, I had multiple reasons to

quit. Through all of those weak, painful, and vulnerable moments, my spouse, Dr. Abhinav Prakash, has reasoned, encouraged, supported, manipulated, and sometimes challenged me to continue. He has given my work equal importance as his own, respected my priorities and choices like his own, persevered to achieve my goals as his own, and sometimes consumed my time as if it is his own. His support and patience played a big role in my pursuit of a PhD. The decision to spend my life with him is the wisest decision I made, I am extremely proud of it. I am excited to see what the future holds for us.

B.J. Neblett said “We are sum total of our experiences.” Well, I disagree. We are sum total of our experiences and the experiences of the people that surround us. I cannot even begin to comprehend how my life has been influenced by my family. I am most grateful to my family for my upbringing and for inculcating the values that I hold profoundly. Udhavdas Kokel’s experiences of the 1947 partition and the dislocation of our Sindhi community inculcated a deep sense of community service and tireless hard work in him and then in me, his granddaughter. Divya Kokel’s relentless attitude towards handling the never-ending workload and emergencies has given me, her daughter, the confidence to weather any storm with tenacity. These values—community service, hard work, and relentless attitude—have helped me fulfill my job as a PhD student with utmost honesty. Thank you, Mummy (Ahmedabad), Heena didi, Mohit, Amma, Bharatu Kaka, and Mummy Patna for your love, support, and encouragement. And, for shielding me from the burden of responsibilities. I thank the Kokel family, the Kaku family, Ketan, and Kartik for being there. My close friends, in Dallas and around the globe, have been the power bank when I needed a recharge. Thank you, Vidhi and Adrita, for being my family in this foreign land. Most importantly, I would like to thank Narendra Kokel, my father, who in his calm and unobtrusive way provided for and supported all my aspirations. How I wish he could have been here to see them come true. I miss him dearly.

I thank the staff of the computer science department at The University of Texas at Dallas for helping me every semester, especially, Mr. Douglas Hyde. He entertained all my requests on time and often provided great advice in his charismatic manner. I acknowledge the support of DARPA CwC Program Contract W911NF-15-1-0461, ARO award W911NF2010224, and AFOSR award FA9550-18-1- 0462. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFOSR, ARO, DARPA, or the U.S. government.

February 2023

BEYOND DATA:  
EFFICIENT KNOWLEDGE-GUIDED LEARNING  
FOR SPARSE AND STRUCTURED DOMAINS

Harsha Kokel, PhD  
The University of Texas at Dallas, 2023

Supervising Professor: Sriraam Natarajan, Chair

The field of AI has made great advances in recent years. Most of these advances have focused on leveraging more data and finding new architectures to improve system performance. However, collecting data can lead to exorbitant costs. This is especially the case for structured domains where the data conforms to some standardized format (like tabular data, relational databases, etc.). In structured domains, an expert might be required to collect and organize data; necessitating time and effort. Further, learning explicitly from data is neither sufficient nor favorable. Enormous data can cause concerns for safety, lack of fairness, and a substantial carbon footprint. So looking beyond learning from data, this dissertation focuses on *finding principled ways to leverage rich human knowledge for sparse and structured domains to guide the learning procedure*.

In particular, this dissertation looks at *four* challenges that arise when models are learned in structured domains and propose to tackle them using explicit human knowledge. **First**, we consider the challenge of learning from sparse and noisy data in the successful gradient boosting framework and propose to use domain-specific trend information to improve prediction. **Second**, we consider the challenge of learning to generalize across multiple tasks and objects in sequential decision making. We address this challenge by proposing a framework that

takes inspiration from human’s ability to generalize by identifying compositionality and generating abstract representations. **Third**, we consider the challenging task of human-machine collaborative problem solving and propose a framework that uses natural language communication for effective bi-directional interaction. Finally, the **fourth** challenge we consider is the problem of a large hypothesis space when dealing with domains with heterogeneous objects. We identify the lack of a language bias—typed object representations—in recent neurosymbolic architectures and devise an approach to incorporate the bias.

In this dissertation, we demonstrate various ways to incorporate domain-specific knowledge from humans in training AI systems. We conclude that using domain knowledge not only reduces the sample complexity but also improves the performance and generalization abilities of the model.

## TABLE OF CONTENTS

ACKNOWLEDGMENTS . . . . .	v
ABSTRACT . . . . .	ix
LIST OF FIGURES . . . . .	xvii
LIST OF TABLES . . . . .	xxii
CHAPTER 1 INTRODUCTION . . . . .	1
1.1 Motivational example . . . . .	1
1.1.1 Problem 1: Sparse and noisy data . . . . .	2
1.1.2 Problem 2: Multi-task learning and generalization . . . . .	3
1.1.3 Problem 3: Collaborative problem solving . . . . .	3
1.1.4 Problem 4: Large hypothesis space . . . . .	4
1.2 Dissertation overview . . . . .	5
1.2.1 Dissertation statement . . . . .	5
1.2.2 Dissertation contributions . . . . .	5
1.2.3 Dissertation outline . . . . .	6
CHAPTER 2 TECHNICAL BACKGROUND . . . . .	10
2.1 Structured data . . . . .	10
2.1.1 Tabular . . . . .	10
2.1.2 First-order logic . . . . .	11
2.2 Gradient-boosted trees . . . . .	13
2.3 Qualitative influence information . . . . .	15
2.3.1 Monotonic influence . . . . .	16
2.3.2 Synergistic influence . . . . .	17
2.3.3 QI for decision-making . . . . .	17
2.4 Sequential decision making . . . . .	18
2.4.1 Reinforcement learning . . . . .	18
2.4.2 Planning . . . . .	24
2.5 Statistical relational AI . . . . .	27
2.5.1 First-order conditional influence language . . . . .	29

2.6	Neurosymbolic AI . . . . .	30
PART I SPARSE AND NOISY DOMAINS . . . . .		34
CHAPTER 3 INCORPORATING QUALITATIVE INFLUENCE INFORMATION		35
3.1	Introduction . . . . .	35
3.2	Related work . . . . .	37
3.3	Knowledge-intensive gradient boosting . . . . .	38
3.3.1	Monotonic constraint . . . . .	39
3.3.2	Interpretation of the update equation . . . . .	43
3.3.3	Equilibrium between advice and data . . . . .	44
3.3.4	Overfitting by strict monotonicity . . . . .	46
3.3.5	KiGB algorithm . . . . .	47
3.3.6	Classification . . . . .	48
3.3.7	Extensions . . . . .	48
3.4	Experiments . . . . .	49
3.4.1	Datasets . . . . .	49
3.4.2	Standard gradient boosting baselines . . . . .	51
3.4.3	Monotonic gradient boosting baselines . . . . .	51
3.4.4	Robustness to the hyperparameters . . . . .	53
3.4.5	Real data sets . . . . .	55
3.4.6	Learning curve . . . . .	56
3.5	Summary . . . . .	57
PART II MULTI-TASK LEARNING AND GENERALIZATION . . . . .		59
CHAPTER 4 INTEGRATING RELATIONAL PLANNING AND REINFORCEMENT LEARNING . . . . .		60
4.1	Introduction . . . . .	60
4.2	Related work . . . . .	62
4.2.1	Planner and RL combination . . . . .	62
4.2.2	Abstraction . . . . .	63
4.3	Relational planning and Learning . . . . .	63

4.3.1	Motivational example . . . . .	63
4.3.2	Problem setup . . . . .	66
4.3.3	RePreL architecture . . . . .	67
4.3.4	D-FOCI . . . . .	70
4.3.5	Example of D-FOCI . . . . .	71
4.3.6	Abstraction using D-FOCI . . . . .	72
4.3.7	Learning . . . . .	76
4.4	Experiments . . . . .	76
4.4.1	Domains . . . . .	77
4.4.2	Sample efficiency . . . . .	80
4.4.3	Task transfer . . . . .	81
4.4.4	Generalization . . . . .	83
4.5	Conclusion . . . . .	85
CHAPTER 5 REPREL EXTENSION TO DEEP, NEURAL RL . . . . .		88
5.1	Deep RePreL . . . . .	88
5.1.1	Batch learning . . . . .	89
5.1.2	Recursive abstraction with DRRL . . . . .	91
5.2	Experiments . . . . .	92
5.2.1	Deep RL . . . . .	93
5.2.2	Deep relational RL . . . . .	94
5.3	Conclusion . . . . .	98
CHAPTER 6 REPREL EXTENSION FOR HYBRID DATA . . . . .		102
6.1	Introduction . . . . .	102
6.2	Hybrid deep RePreL . . . . .	103
6.3	Experiments . . . . .	108
6.3.1	Domains . . . . .	108
6.3.2	Baselines . . . . .	109
6.3.3	Sample efficiency . . . . .	110
6.3.4	Generalization . . . . .	110

6.4	Conclusion . . . . .	111
PART III COLLABORATIVE PROBLEM SOLVING . . . . .		114
CHAPTER 7 PLANNING AND LEARNING VIA COMMUNICATION . . . . .		115
7.1	Introduction . . . . .	115
7.2	Preliminaries . . . . .	117
7.2.1	Concept learning . . . . .	117
7.2.2	Neural parsers . . . . .	118
7.2.3	Minecraft . . . . .	118
7.3	Lara - Planning and learning via communication . . . . .	118
7.3.1	Problem definition . . . . .	119
7.3.2	System setup . . . . .	121
7.3.3	System architecture . . . . .	124
7.3.4	Minecraft simulator . . . . .	124
7.3.5	NLP engine . . . . .	125
7.3.6	Planner . . . . .	127
7.3.7	Concept learner . . . . .	129
7.4	Demonstration . . . . .	133
7.5	Related work . . . . .	133
7.6	Conclusion . . . . .	134
PART IV LARGE HYPOTHESIS SPACE . . . . .		136
CHAPTER 8 LANGUAGE BIAS IN NEUROSymbolic MODELS . . . . .		137
8.1	Background . . . . .	137
8.1.1	ILP task . . . . .	137
8.1.2	Biases in ILP . . . . .	138
8.1.3	Biases in NeSy . . . . .	139
8.1.4	NeSy predicate invention . . . . .	141
8.2	Introducing Typed Bias . . . . .	143
8.2.1	Approach I . . . . .	143
8.2.2	Approach II: HTRI . . . . .	144

8.3	Experiments . . . . .	144
8.4	Discussion . . . . .	146
PART V OTHER EXPLORATIONS . . . . .		148
CHAPTER 9 EXTRACTING QUALITATIVE KNOWLEDGE . . . . .		149
9.1	Introduction . . . . .	149
9.2	Extracting qualitative influences . . . . .	150
9.3	Evaluation on nuMoM2b study . . . . .	152
9.3.1	The nuMoM2b study . . . . .	152
9.3.2	Setup and baselines . . . . .	154
9.3.3	Results . . . . .	154
9.4	Summary . . . . .	155
CHAPTER 10 DATA SUBSET SELECTION FOR DOMAIN ADAPTATION . . . .		158
10.1	Introduction . . . . .	158
10.2	Related work . . . . .	161
10.2.1	SDA . . . . .	161
10.2.2	Subset selection methods . . . . .	162
10.3	Preliminaries . . . . .	163
10.3.1	Submodular functions . . . . .	163
10.3.2	Submodular mutual information . . . . .	164
10.3.3	SDA loss functions . . . . .	164
10.4	ORIENT . . . . .	165
10.4.1	Algorithm . . . . .	167
10.5	Connections to previous work . . . . .	168
10.6	Experimental evaluation . . . . .	169
10.6.1	Comparison of different SMI functions . . . . .	174
10.7	Summary . . . . .	175
CHAPTER 11 CONCLUSION AND FUTURE WORK . . . . .		176
PART VI APPENDIX . . . . .		180
APPENDIX A KNOWLEDGE-INTENSIVE GRADIENT BOOSTING . . . . .		181
A.1	Why gradient boosted trees? . . . . .	181

APPENDIX B	REPREL	186
B.1	Traditional relational RL	186
APPENDIX C	LARA	187
C.1	Dialogue manager	187
C.2	Background file	189
C.3	Planner	199
C.4	Concept Learning	200
APPENDIX D	ORIENT	203
D.1	Concave over modular mutual information	203
D.2	Proof of the theorems	203
D.3	SDA loss	206
D.4	Experiment details	208
D.5	Additional results	210
D.5.1	Synthetic experiments	213
D.5.2	Analysis of data subset size	214
D.5.3	Analysis of $L$ for subset selection	215
D.5.4	Analysis of time taken	215
REFERENCES		224
BIOGRAPHICAL SKETCH		246
CURRICULUM VITAE		

## LIST OF FIGURES

1.1	An illustrative autonomous food delivery system. . . . .	2
2.1	An entity-relationship diagram of Taxi domain. The domain has 2 types of entities, passenger and location. Passengers have two boolean attributes: <b>at_dest</b> and <b>in_taxi</b> . Locations have two real attributes <b>x_coord</b> and <b>y_coord</b> . Both entities are related by two relations: <b>at</b> and <b>dest</b> . <b>at</b> relation indicates represents the current location of the passenger and <b>dest</b> location represents the final destination of the passenger. . . . .	12
2.2	Illustration of monotonic influences between two random variables $X$ and $Y$ . (a) Isotonic influence (b) Antitonic influence . . . . .	16
2.3	Illustration of an RL framework. . . . .	19
2.4	Comparison of actions in MDPs, SMDPs, and HRL . . . . .	21
2.5	Example of a hierarchical planning task consisting of methods, operators, initial state, and goal condition. . . . .	26
2.6	Statistical Relational AI (StarAI) combines Logic, Probability, and Learning. . .	28
2.7	(a) An instance of a domain with all the objects. (b) Grounded Bayesian network obtained for the FOCI statement in Equation 2.15 . . . . .	30
2.8	Taxonomy of Neurosymbolic approaches. . . . .	33
3.1	An example to illustrate the need for equilibrium between data and advice. (a) A toy dataset. The horizontal axis represents the feature variable $A$ , the vertical axis represents the feature variable $B$ , and markers represent the target variable $Y$ . An expert provides a monotonic influence statement $A \stackrel{Q^+}{\prec} Y$ . Region R1 and R2 violate the advice. (b) Illustration of the decision boundary learned by standard gradient-boosting approach. (c) Illustration of decision boundary learned by monotonic gradient-boosting approach. . . . .	45
3.2	Illustration of the decision boundaries learned by KiGB approach with different relative importance to advice (different $\lambda$ values). . . . .	45
3.3	Illustration of the overfitting by LMC. As can be seen, LGBM, without any monotonic influence statements, learned an incorrect model due to the presence of noisy data. With LMC, the model learns a monotonic function but it overfits the training data. LKiGB provides a correction to the LGBM and generalizes to a better model. . . . .	46
3.4	Analysis of sensitivity to hyperparameters: $\lambda$ & $\varepsilon$ . (a) Comparison of accuracy with SGB and MONO for a classification task. (b) Comparison of negative mean-squared error with LGBM and LMC for a regression task. The higher the better. . . . .	54
3.5	Learning curve for classification task in HELOC dataset with KiGB, standard boosting, and monoensemble . . . . .	57

4.1	Motivational example of RePreL framework in Taxi domain. ① Relational taxi domain. The initial state has two passengers $p1$ and $p2$ . The goal is to transport both of them to their respective destination, $d1$ and $d2$ respectively. ② The high-level planner decomposes the goal into subgoals by using high-level state representation. ③ Low-level RL agents achieve these subgoals by using abstract state representations. . . . .	65
4.2	RePreL architecture. . . . .	67
4.3	(a) Craft World indicating domain with eight locations: grass, wood, iron, gold, gem, workbench, toolshed, and factory. Black cells in the grid represent walls. (b) Office World, reproduced from Illanes et al. (2020). . . . .	78
4.4	Tasks in the Relational Box World. . . . .	79
4.5	Comparing learning curves of RePreL, TRL, HRL, and QL in Craft World environment. Transferred policies are indicated by “+T”. <i>Task 1</i> is to get wood and iron, <i>Task 2</i> is make stick, <i>Task 3</i> is make axe, <i>Task 4</i> is mine gem. . . . .	82
4.6	Comparing learning curves of RePreL, TRL, HRL and QL in Office World environment. Transfer algorithms are indicated by “+T”. <i>Task 1</i> is to deliver mail, <i>Task 2</i> is to deliver coffee, <i>Task 3</i> is to deliver mail and coffee, and <i>Task 4</i> is to visit A, B, C, D. Note that the RePreL and RePreL+T curves in Task 2 are overlapping. . . . .	83
4.7	Comparing learning curves of RePreL, TRL, HRL, and QL in the Extended Taxi World. <i>Task 1</i> is to drop passenger $p1$ , <i>Task 2</i> is to drop $p1$ and $p2$ , <i>Task 3</i> is to drop $p1$ , $p2$ , $p3$ . . . . .	84
4.8	Comparing learning curves of RePreL and TRL with and without transfer in Box World environment. The goal in all the tasks is to collect the gem, we increase the number of objects to reach the gem in each task. . . . .	84
5.1	RePreL architecture. . . . .	89
5.2	Comparing learning curves of deep RL based learners in the Office World environment. RePreL compared against TRL, HDQN and DQN in (a) <i>Task 1</i> deliver mail and (b) <i>Task 2</i> deliver coffee. Agents are swapped after $1e6$ steps. RePreL+T compared against TRL+T, HDQN+T and DQN+T in (c) <i>Task 1</i> and (d) <i>Task 2</i> . The shaded region depicts the standard deviation. . . . .	94
5.3	Comparing learning curves of deep RL-based learners in the Extended Taxi World. (a) <i>Task 1</i> is to drop passenger $p1$ , (b) <i>Task 2</i> is to drop $p1$ and $p2$ , (c) <i>Task 3</i> is to drop $p1$ , $p2$ , $p3$ . The shaded region depicts the standard deviation. . . . .	95
5.4	FetchBlockConstruction . . . . .	95
5.5	Comparing learning curves of deep relational RL-based learners in the Extended Taxi World. (a) <i>Task 1</i> is to drop passenger $p1$ , (b) <i>Task 2</i> is to drop $p1$ and $p2$ , (c) <i>Task 3</i> is to drop $p1$ , $p2$ , and $p3$ . The shaded region depicts the standard deviation. . . . .	97

5.6	Comparing deep relational RL-based learners in FetchBlockConstruction. (a) Compares learning curve on the task of moving one block (b) Evaluates generalization for moving 1–4 blocks. . . . .	97
6.1	(a) Taxi location and geography information available as an image. (b) Passenger location and destination information as state predicates from a passenger’s mobile.	104
6.2	Proposed HDRePReL architecture. . . . .	105
6.3	Craftworld domain. . . . .	108
6.4	Comparing learning curves of hybrid deep RePReL with DQN in Hybrid Taxi World. (a) <i>Task 1</i> is to drop passenger p1, (b) <i>Task 2</i> is to drop p1 and p2, (c) <i>Task 3</i> is to drop p1, p2, p3. . . . .	110
6.5	Comparing learning curves of HDRePReL with DQN in Craft World. (a) <i>task 1</i> is to make bread; (b) <i>task 2</i> is to build house; (c) <i>task 3</i> is to break rock. . . . .	111
7.1	(a) Minecraft builder screen showing the 3D build region and the chat interface. (b) Example of a target structure in the oracle screen. The architect can see both screens. . . . .	119
7.2	Target structure on the left is visible only to the architect (A). The architect instructs the builder (B) to build a red tower. B seeks clarification about the size and then proceeds to build the tower in the build region. . . . .	121
7.3	(a) Five of the eight primitive structures: tower, row, column, cube, and cuboid. Block indicators that point to a single block of the structure. (b) top-end and bottom-end, (c) left-end and right-end, (d) back-end and front-end. For complex structures, these indicators can be combined, for example, the front-bottom-left block of a cube. . . . .	122
7.4	Architecture of Lara. It consists of a Minecraft simulator, NLP engine, planner, and concept learner. An example flow of building the “L” shape is illustrated. ① Natural language instruction by the architect to build a red tower. ② AMR representation of the instruction parsed by AMR parser. ③ Logic representation of the instruction. ④ Block placement plan generated by the planner. ⑤ Agent control executes the plan in the build region. ⑥ Next instruction from the architect to add a row. ⑦ Agent control executes the next instruction in the build region. ⑧ Structure saved as “L”. . . . .	123
7.5	FOL representation of collaborative building task. (a) FOL language (Complete list of predicates is deferred to Appendix) (b) Example FOL instructions. . . . .	125
7.6	(a) Goal description of the FOL instruction. (b) FOL representation of new structure “L” and active advice queries to the architect. . . . .	128

7.7	Illustration of Lara. (a) A new mission—red tower of height 4. (b) Greeting from the builder and first instruction from the architect to build a red tower. (c) The builder enquires about the height of the tower and the architect provides the height value 4. (d) The builder understands the task, generates a plan, and placed red blocks in the build region. (e) After completing the instruction the builder asks for the next instruction and the architect says done. This ends the mission. (f) The complete chat of this mission is here in text for clarity ‘<B>’ and ‘<A>’ indicates a message from the builder and the architect, respectively. . . .	131
7.8	Demonstration of the concept hierarchy. (a) Gamma structure ( $\Gamma$ ) made from a row and column. (b) Cap structure ( $\sqcap$ ) made from gamma and a column. (c) Box structure ( $\square$ ) made from a cap and a row. Subfigures (d), (e), and (f) present the natural language instructions. . . . .	132
8.1	HRI architecture. . . . .	142
8.2	Learning curves comparing precision and recall, of HRI with HTRI and Masked-HRI, for learning operator preconditions. . . . .	146
8.3	Learning curves comparing precision and recall, of Masked-HRI and Masked(T)-HRI, for learning operator preconditions. . . . .	147
9.1	Causal network obtained by the stable PC algorithm for the nuMoM2b dataset.	154
10.1	Illustration of ORIENT framework. ① Given the target data $D^t$ and source data $D^s$ , a subset $A \subseteq D^s$ and model parameters $\theta$ are randomly initialized. ② Model parameters $\theta$ are optimized for $L$ epochs on the subset $A$ with any gradient descent (GD) based method. ③ Subset $A$ is updated using the SMI measure and current model parameters $\theta$ after every $L^{th}$ epoch. ④ Model is trained for $E$ epochs, with subset selection after every $L$ interval. ⑤ The final model parameters are returned after $E$ epochs. . . . .	165
10.2	Speed up vs prediction accuracy on three domains of Office31 dataset: Amazon (A), DSLR (D), and Webcam (W). $X$ -axis represents the speed up by the model, i.e. the ratio of the time taken to train on complete source dataset (Full) to the time taken by the model. $Y$ -axis represents the prediction accuracy of the model on the target domain.	170
10.3	Convergence curves on four domains of Office-Home dataset: Art (A), Clipart (C), Product (P), and Real World (R). $X$ -axis presents the training time in hours and $Y$ -axis presents the prediction accuracy on the target domain. . . . .	172
10.4	Speed up achieved by combining d-SNE with ORIENT . . . . .	173
10.5	GradCam (Selvaraju et al., 2017) activation maps of the models learned using d-SNE + Full and d-SNE + ORIENT (FLMI) on the Office-Home dataset with “Product” as the source domain and “Real World” as the target domain. As evidenced by class activation maps, the ORIENT framework enabled the model to learn more effective class discriminative features than Full data training. . . .	173

A.1	Comparison of linear support vector regression (SVR-L), linear regressor with L1 prior as a regularizer (Lasso), support vector regressor with radial basis function kernel (SVR-K), bagging ensemble of decision trees (Bagging), deep neural network estimator (DNN), and gradient-boosted decision trees (GBT) for the task of predicting the shipment prices in Logistics dataset. Y-axis presents the mean absolute percentage error (MAPE) in prediction. . . . .	182
C.1	Example of concepts. . . . .	201
D.1	GradCam (Selvaraju et al., 2017) activation maps of the models learned using $d$ -SNE + Full and $d$ -SNE + ORIENT (FLMI) on the Office-Home dataset with “Product” as the source domain and “Real World” as the target domain. As evidenced by class activation maps, the ORIENT framework enabled the model to learn more effective class discriminative features than Full data training consistently.	209
D.2	Subsets selected by different instantiations of ORIENT on a synthetic dataset. (a) Synthetic data - We sample 50 examples from the target distributions for the query set. (b) ORIENT (FLMI) selects samples close to the target distribution. (c) ORIENT (GM) selects representative samples from the source domain. (d) ORIENT (G) selects samples near the decision boundaries of the source domains. (e) ORIENT(LDMI) prioritizes selection of data samples from certain classes over others (f) ORIENT(GCMI) selects samples from a source domain that are very similar and clustered together. . . . .	220
D.3	Subsets selected by different instantiations of ORIENT on a synthetic dataset. (a) Synthetic data - We sample 50 examples from the target distributions for the query set. Target domain is skewed to the right for class 0 and left for class 1 as compared to the source domain. (b) ORIENT (FLMI) selects samples close to the target distribution. (c) ORIENT (GM) selects representative samples from the source domain. (d) ORIENT (G) selects samples near the decision boundaries of the source domains. (e) ORIENT(LDMI) prioritizes selection of data samples from certain classes over others (f) ORIENT(GCMI) selects samples from a source domain that are very similar and clustered together. . . . .	221
D.4	Convergence curves on four domains of Office-Home dataset: Art (A), Clipart (C), Product (P), and Real World (R). X-axis presents the training time in hours and Y-axis presents the Validation loss on the target domain. . . . .	222

## LIST OF TABLES

3.1	Datasets used in the experiments. The first 5 datasets have binary classification tasks, the next 10 have regression tasks and the last two are real-world datasets described in Section 3.4.5. The second column either refers to the literature from where we got the monotonic features and/or lists the feature names used for experiments. Features in <b>bold</b> have negative influence ( $X \stackrel{Q}{\prec} Y$ ) and others have positive influence ( $X \stackrel{Q}{\succ} Y$ ). . . . .	50
3.2	Standard baselines: Comparison of performance of SKiGB and SGB. The performance measure used is accuracy for classification tasks (the higher the better) and mean squared error for regression tasks (the lower the better). . . . .	52
3.3	Monotonic baselines: Comparison of accuracy of KiGB and monotonic boosting approaches for classification tasks. . . . .	52
3.4	Monotonic baselines: Comparison of mean squared error of LKiGB and LMC for regression tasks. . . . .	52
3.5	Comparison of KiGB with standard and monotonic baselines on real-world datasets.	55
4.1	Illustrative example of recursive unrolling of the D-FOCI statements in taxi-domain.	73
4.2	D-FOCI statments and relevant state predicates for all the domains . . . . .	87
5.1	Summary of hyperparameters used in deep RL experiments (Section 5.2.1). . . .	99
5.2	Summary of hyperparameters used in deep relational RL experiment of Extended Taxi World (Section 5.2.2). . . . .	100
5.3	D-FOCI statements and relevant features (literals) of the state that form the abstract state. . . . .	101
6.1	Summary of the network hyperparameters . . . . .	113
8.1	Summary of the tasks . . . . .	145
9.1	Summary of the 8 variables in nuMoM2b dataset. . . . .	153
9.2	Comparision of QI from prior knowledge (PK), QuaKE, and Data Alone. $\checkmark/\times$ represents that this relationship does/not exist respectively while $?$ represents unknown influence. The three groups of rows show (1) MI, (2) SI, and (3) sub-SI. Colors highlight rules recovered by QuaKE and show (a.) coherent with the PK and baseline in white (b.) contradicting the baseline in green (c.) coherent with baseline but contradict the PK in blue. . . . .	157
10.1	Instantiations of submodular mutual information functions. . . . .	166
10.2	Test accuracy for office-31 with SDA methods . . . . .	171
10.3	Test accuracy for Office-Home with SDA methods . . . . .	171

A.1	List of features used for the shipment price prediction task on logistics data. . .	183
D.1	Test accuracy on Office-31 dataset. . . . .	210
D.2	Training time (in hours) for 300 epochs on Office-31 dataset . . . . .	210
D.3	Test accuracy on Office-31 dataset with SDA methods . . . . .	211
D.4	Training time in hours on Office-31 with SDA methods . . . . .	211
D.5	Test accuracy on Office-Home dataset . . . . .	212
D.6	Training time(in hours) for 300 epochs on Office-Home dataset . . . . .	212
D.7	Test accuracy on Office-Home with SDA methods . . . . .	212
D.8	Training time(in hours) on Office-Home with SDA methods . . . . .	212
D.9	Comparison of test accuracy for office-31 with d-SNE loss function and different fractions of subset selection. . . . .	214
D.10	Training time in hours on Office-31 with d-SNE loss function and different fractions of subset selection. . . . .	214
D.11	Table showing target domain accuracy and training time taken(in hrs) achieved on office-31 ( $A \rightarrow D$ ) using d-SNE loss function and 30% subset. . . . .	215
D.12	Setting: $R \rightarrow P$ . . . . .	216
D.13	Setting: $R \rightarrow C$ . . . . .	216
D.14	Setting: $R \rightarrow A$ . . . . .	216
D.15	Setting: $P \rightarrow R$ . . . . .	217
D.16	Setting: $P \rightarrow C$ . . . . .	217
D.17	Setting: $P \rightarrow A$ . . . . .	217
D.18	Setting: $C \rightarrow R$ . . . . .	218
D.19	Setting: $C \rightarrow P$ . . . . .	218
D.20	Setting: $C \rightarrow A$ . . . . .	218
D.21	Setting: $A \rightarrow R$ . . . . .	219
D.22	Setting: $A \rightarrow P$ . . . . .	219
D.23	Setting: $A \rightarrow C$ . . . . .	219
D.24	Speed ups achieved by different variants of ORIENT when using $1.05 \times$ the minimum validation loss as a stopping criterion for training. . . . .	223

# CHAPTER 1

## INTRODUCTION

The field of Artificial Intelligence (AI) has made great advances in recent years that have led to the adoption of AI in various industrial use cases. While increasing model size, training on bigger and more diverse datasets, and supplementing computational abilities may advance AI in certain domains, we need further advancement for generalized intelligence and broader adoption. Learning exclusively from data is neither sufficient nor favorable. Relying on enormous data can cause concerns for safety, lack of fairness, inadequate representation of the target population, etc, and training on large datasets results in a substantial carbon footprint. Additionally, many problems have limited data and collecting more training data may lead to exorbitant costs. This is especially the case for structured data, where a domain expert might be required to annotate or collect data in a structured form. *In this thesis, we look beyond learning from data and focus on finding principled ways to leverage human knowledge for domains that have sparse and structured data.* We aim to find ways to guide, advice, and instruct domain-specific knowledge to AI approaches and make them sample efficient.

We look at several problems that arise when learning is performed exclusively from data and propose to tackle them using explicit human knowledge. Consider the illustrative example of an autonomous food delivery system in Figure 1.1.

### 1.1 Motivational example

An autonomous food delivery system, shown in Figure 1.1, has to provide food offerings, coordinate with restaurants, estimate delivery time, process orders, schedule deliveries, plan routes, deliver orders, provide personalized recommendations, etc. It consists of various components that help in different tasks. For example, estimating delivery time requires some

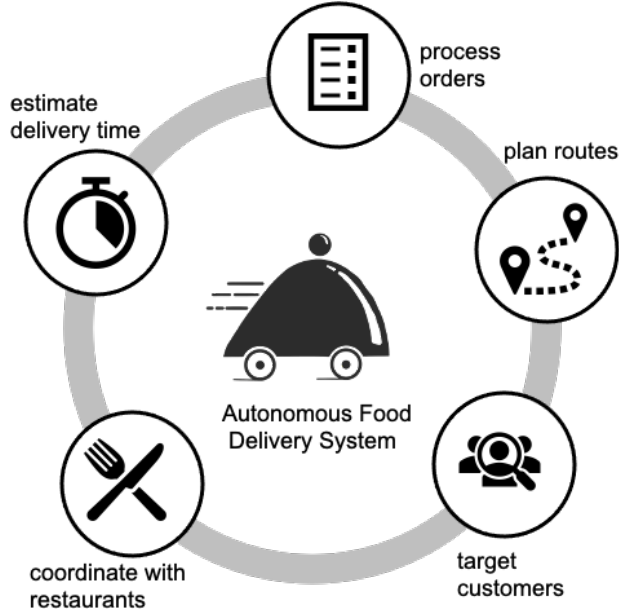


Figure 1.1: An illustrative autonomous food delivery system.

predictive modeling, scheduling orders and planning routes require automated sequential decision making, delivering order requires the ability to communicate with humans, and a successful recommendation system require efficient learning of user preferences. It is very difficult to rely only on data for all of these individual components.

### 1.1.1 Problem 1: Sparse and noisy data

Delivery time estimates obtained only using the historic order data may be inadequate for a queried location  $L$  if location  $L$  is not seen before. Historic data might be sparse, biased, imbalanced, and noisy. There might not be any orders delivered to location  $L$  before, but if it is known that location  $L$  is beyond location  $K$  then the model should regard this information and estimate a delivery time that is greater than the delivery time estimate for location  $K$ . Domain-specific information such as the delivery time estimate increases if the location is further away can be captured as a monotonic influence statement (also called qualitative influence information). **We consider the problem of using monotonic**

influence information in the successful gradient boosting framework and develop a unified framework, for both classification and regression settings, to improve prediction in noisy and sparse regions of the sample space.

### 1.1.2 Problem 2: Multi-task learning and generalization

Humans have an innate capability to generalize from one or a small set of examples. Consider the example of driving. Humans can easily generalize from driving in one neighborhood to driving in another. An autonomous delivery system must also generalize similarly, without using an exorbitant amount of data. In addition, humans leverage compositionality and task-specific representations to generalize and transfer skills across various tasks. However, sequential decision-making problems in multi-task domains are challenging for current AI systems. Especially so in relational settings where data can have a varying number of entities and relations between those entities. **We consider the problem of sequential-decision making in relational domains, with multiple tasks. We propose a hierarchical framework called RePReL that integrates AI planning and reinforcement learning to learn transferable and generalizable skills.** The planner works as a deliberate, rational, symbolic reasoning system that decomposes the task into sub-tasks. RL agents at the lower-level act as reactive, goal-oriented policies that solve each sub-task independently. We provide a knowledge-based approach for task-specific state abstractions that enables efficient skill learning.

### 1.1.3 Problem 3: Collaborative problem solving

Consider the situation when the person delivering our order does not arrive or is lost. In such a situation, we would call them and try to figure out where they are. We communicate and collaborate with them to design a plan that leads them to the correct location. An autonomous delivery system might get lost as well, so it is important to support human

interaction. With effective communication, humans might be able to determine where the autonomous agent is lost and collaborate in order to get the order delivered to the preferred location. However, human-machine communication and collaborative problem-solving are quite challenging tasks. They require bi-directional communication, shared perception of the world, sophisticated language parser, and contextual understanding. **We consider the problem of human-machine collaborative problem solving as a planning task coupled with natural language communication. We propose a framework that not only collaborates to solve the problem but also induces new, rich concepts based on limited human interaction.** We illustrate the ability of this framework on a collaborative building task in a Minecraft-based blocks world domain.

#### 1.1.4 Problem 4: Large hypothesis space

Finally, when the hypothesis space is large, various biases are used to prune the space. Deep learning approaches use implicit biases in architecture. Whereas, symbolic inductive logic programming methods learn a generalizable hypothesis from a few examples of structured data by explicitly declaring the inductive biases in the background knowledge. This enables the specification of strong domain-dependent biases. With the increasing interest in combining neural and symbolic approaches, many neural architectures are proposed to solve symbolic reasoning tasks. However, symbolic, structured data is more difficult to obtain, and hence, we need ways to prune the hypothesis space without using large amounts of data. While many inductive biases in deep learning are already encoded in the various architectures and the message-passing algorithms, more biases are required to further prune the hypothesis space. Especially in the case of structured data. We look at symbolic methods for inspiration on biases for structured data. **We investigate the biases exercised by the neural deep learning and symbolic ILP methods, and highlight a declarative language bias that is missing from the neural approaches. We propose a novel way to implicitly incorporate that declarative bias in a neuro-symbolic architecture.**

## 1.2 Dissertation overview

The central thesis and contributions of this dissertation are as follows.

### 1.2.1 Dissertation statement

Domain knowledge not only helps AI models in case of sparse data but also improves generalizability. Our objective is to identify how can we effectively incorporate different types of domain-specific knowledge while learning to improve efficiency and generalization abilities.

### 1.2.2 Dissertation contributions

- I. Developed a method and approach (KiGB) to effectively incorporate qualitative influence information in the successful gradient-boosting framework for efficient learning with sparse and noisy data.
- II. Proposed and developed a framework (RePReL) for the successful fusion of high-level symbolic reasoning with lower-level signal-based reasoning.
- III. Proposed a method to effectively leverage different levels and kinds of abstraction for task transfer and generalization across objects.
- IV. Proposed a formal language (D-FOCI) to express conditional influence between state variables.
- V. Developed an approach for efficient sequential decision-making with multi-modal data where the data could arrive from multiple sources.
- VI. Proposed and developed a framework (Lara) for collaborative problem-solving via communication.

VII. Developed a framework for incorporating language bias while learning a neuro-symbolic model.

### 1.2.3 Dissertation outline

This dissertation is divided into five high-level parts. The four problems discussed above form four parts of this dissertation and an additional part presents other explorations. The dissertation is organized as follows:

**Chapter 2** presents the required background for various problems addressed in this dissertation. We start with a brief introduction of the structured data, which is the focus of this dissertation. Subsequently, we discuss the preliminaries on the gradient boosted trees and qualitative influence information, which are adapted in Chapter 3. Next, we discuss sequential decision making and its two fields: reinforcement learning and planning. We briefly explain the hierarchical approaches in both these fields. Next, we present the field of Statistical relational AI and explain first-order conditional influence language. These discussions are further extended in Chapters 4–7.

## Part I: Sparse and noisy domains

**Chapter 3** presents our approach to addressing sparse and noisy data problems (Problem 1). We introduce a novel framework to incorporate monotonic influence in the successful gradient boosting framework. The proposed framework is easily applicable for both types of predictive modeling: classification and regression. We empirically evaluate the framework for performance, sample efficiency, and robustness to hyperparameters on different benchmark datasets as well as real-world datasets.

## Part II: Multi-task learning and generalization

**Chapter 4** presents our approach for multi-task learning and generalization problems (Problem 2). We present a novel framework called RePReL that integrates relational planning and reinforcement learning to learn transferable skills by leveraging the compositionality of the domain. We introduce a first-order language called D-FOCI for expressing task-specific influence and propose to use this influence information for task-specific abstraction. We empirically evaluate the proposed framework on 4 different grid world domains.

**Chapter 5** extends the RePReL to deep, neural RL for continuous domains. We extend it by proposing a batch-learning approach that can be used with an off-policy deep RL method. The use of batch learning allows for deep relational RL agents as base learners. This allows us to relax one of the key restrictions for abstraction used in Chapter 4. We empirically evaluate the approach with two different deep RL-based learners, including a deep relational RL approach, for sample efficiency, task transfer, and generalization.

**Chapter 6** extends the RePReL framework for domains with multi-modal data, where data is a fusion of information from multiple sources. Multi-modal data is generally a hybrid collection of structured and unstructured data. To address this, we introduce a novel hybrid deep RePReL framework that uses an input-preprocessing module for unstructured data and a merge module to concatenate structured and unstructured data. We empirically evaluate the proposed approach on two domains for sample efficiency and generalization.

### **Part III: Collaborative problem solving**

**Chapter 7** presents our approach to collaborative problem solving via communication (Problem 3). In this chapter, we focus on the problem of human-machine collaboration for a building task in a Minecraft environment. The collaborative building task requires humans and machines to interact via a chat interface for successful completion. We present

the challenges in this task and propose an integrated system called Lara that uses various modules to effectively use human knowledge to achieve the task and learn new structures.

## **Part IV: Large hypothesis space**

**Chapter 8** investigates approaches to bias neurosymbolic models for domains with large hypothesis space. We present the existing biases in the neurosymbolic frameworks and identify the lack of specific bias—typed object representations. We propose two approaches to incorporate this bias and empirically evaluate it against the current approaches.

**Part V: Other explorations** While Parts I–IV focuses on answering the question, *how to leverage domain-knowledge to improve learning with limited data?* This part delves into possible approaches when domain knowledge is not available.

**Chapter 9** presents an approach to extract the domain-knowledge, specifically qualitative influence information, from data. We propose an approach called QuaKE that learns the joint distribution of the domain using a graphical model and uses this joint distribution to estimate the two types of qualitative influence: monotonic and synergistic influence. We empirically evaluate our approach on a clinical study for early prediction of adverse pregnancy outcomes—nuMoM2b.

**Chapter 10** considers leveraging data from another related domain. We present an efficient domain adaptation approach using subset selection. We use submodular mutual information functions to identify a subset of source data that is similar to the target data for efficient training. We empirically evaluate our approach on two benchmark datasets for domain adaptation and demonstrate the substantial speed-up and potential of improving the performance by our approach.

**Chapter 11** provides insights into our work presented in the earlier chapters. We present our concluding remarks and discuss some future directions of research.

**Part VI** contains the appendix of the dissertation.

## CHAPTER 2

### TECHNICAL BACKGROUND

This chapter reviews the required backgrounds for the various problems that are addressed in this dissertation. First, Section 2.1 describes structured data and its representations. Then, Section 2.2 and 2.3 describe gradient-boosted trees and qualitative influence information, respectively, which will be adapted in Chapter 3. Then, Section 2.4 describes the background on two approaches for sequential decision-making: reinforcement learning and automated planning, which are integrated and adapted in Chapters 4–6. Finally, Section 2.5 reviews the necessary background of statistical relational AI and dynamic relational probabilistic models.

#### 2.1 Structured data

The data is called structured when it has a well-defined structure, it is in a standardized format that conforms to a model, and each element of the data has an associated semantic interpretable by humans. Contrary to this, unstructured data does not conform to any model, is free form, and hard to organize. Examples of structured data include tabular data, relational databases, graph networks, etc. Examples of unstructured data include images, natural language text, video, audio signals, etc. In this dissertation, we use two major representations for structured data: tabular and first-order predicate logic.

##### 2.1.1 Tabular

The tabular representation of data consists of rows and columns. Each row represents an independent entity and each column represents some attribute of that entity. Since many decision-making tasks (like classification and regression tasks) are concerned with predicting a value for one of these columns, it is common to identify one column in the table as a

‘label’ and refer to the remaining columns as the feature set. We use notation  $\mathbf{x}$  to refer to the feature set and  $y$  to refer to the label. The tabular data with  $N$  rows is represented as  $\{\mathbf{x}, y\}_{i=1}^N$ . Many real-world datasets are represented in tabular format.

Although popular, tabular representation is limited as it requires entities in the data to be independent. It is often used when the underlying data is independent and identically distributed (IID). When the data violates the IID assumption, specifically where the entities are related, the tabular representation is not sufficient.

### 2.1.2 First-order logic

Most businesses collect their data in relational databases. A database can be represented in first-order logic (Lloyd, 1987; Ullman, 1988). First-order logic (or predicate logic) is a rich way of representing data and knowledge. First-order logic representations have been used in AI since the early 1970s. The early expert systems used it for logic programming and automated theorem proving. First-order logic programs use inference rules and axioms to deduce facts for a given first-order theory. First-order logic representation not only allows the following connectives:  $\neg$  (negation),  $\wedge$  (conjunction),  $\vee$  (disjunction),  $\rightarrow$  (implication), and  $\leftrightarrow$  (biconditional)<sup>1</sup>, but also allows quantifiers:  $\exists$  (existential) and  $\forall$  (universal).

A first-order language  $\mathcal{L}$  consists of a finite set of constants, predicates, and free variables. Each predicate takes a fixed number of arguments—arity  $\alpha$ —and is represented as `predicate`/ $\alpha$ . Following the Prolog notation, we represent free variables in uppercase and constants in lowercase. A substitution  $\theta$  maps variables to constants. An atom is a predicate symbol followed by a parenthesized list of terms, `predicate(term1, term2, ...)`. A literal is an atom or negation of an atom. If all the terms in an atom are constants it is called a grounded atom, otherwise, it is called a lifted atom. A lifted atom can be grounded

---

<sup>1</sup>Like the propositional logic

by substitution. For example, substitution  $\theta = \{X/p1\}$  grounds atom  $a = \text{taxi\_at}(X)$  to  $a\theta = \text{taxi\_at}(p1)$ .

## Databases and First-order Logic

Relational databases can be represented in  $\mathcal{L}$ , and vice versa; a first-order logic dataset can be stored in a relational database (Dzeroski, 2010). Datalog, a powerful database query language based on logic programming also uses first-order language (Abiteboul et al., 1995). Consider the Entity-Relationship (ER) diagram shown in Figure 2.1. Passenger and location entities can be represented by constants  $\langle p1, p2, \dots, pn, \rangle$  and  $\langle l1, l2, \dots, ln \rangle$ , respectively. Boolean attributes of passenger, `in_taxi` and `at_dest`, can be represented using unary predicates `in_taxi/1` and `at_dest/1` (for instance, `in_taxi(p1)`). Attributes of location, `x_coord` and `y_coord` can be represented using binary predicate `x_coord/2` and `y_coord/2` (for example, `x_coord(l1, 0.1)`). Relationships `at` and `dest`, between the passenger and location, can be represented with binary predicates `at/2` and `dest/2` (for instance, `at(p1, l1)`).

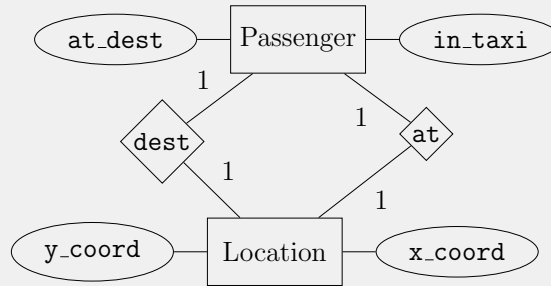


Figure 2.1: An entity-relationship diagram of Taxi domain. The domain has 2 types of entities, passenger and location. Passengers have two boolean attributes: `at_dest` and `in_taxi`. Locations have two real attributes `x_coord` and `y_coord`. Both entities are related by two relations: `at` and `dest`. `at` relation indicates represents the current location of the passenger and `dest` location represents the final destination of the passenger.

## 2.2 Gradient-boosted trees

Gradient Boosted Trees have significantly outperformed other ML algorithms for various tabular and structured datasets. A study by Olson et al. (2018) compared 13 state-of-the-art ML algorithms on 165 publically available datasets and ranked gradient-boosted trees as the best. We use gradient-boosted models in Chapter 3, so this section provides the required gradient-boosting background.

### Boosting

Boosting (Freund and Schapire, 1996) is an ensemble learning approach where multiple “weak” models are learned to produce a “strong” model. A model is called **weak** when the prediction of the model is only slightly better than random guessing. In boosting, weak models are learned sequentially on weighted examples. In each round higher weight is given to examples that have larger prediction errors. This forces the new weak model to concentrate on improving the prediction of the previous model. The final prediction is made by combining the predictions of all the models. A major advantage of forming a strong model using boosting is that such models have significantly low variance (as compared to learning a single strong model). Boosting is also interpreted as a gradient descent algorithm in function space (Breiman, 1999; Mason et al., 1999; Friedman, 2001). Friedman (2001) extends this view and proposes a generic functional gradient boosting algorithm for different loss functions.

### Gradient Descent

The objective of a parameter estimation problem is to estimate the parameters ( $\mathbf{w}$ ) of a function ( $F$ ) that best fit the dataset  $(\{\mathbf{x}_i, y_i\}_{i=1}^N)$ . “Best fit” is often measured by a loss

function ( $\mathcal{L}$ ), so the function estimation objective is as follows,

$$\arg \min_{\mathbf{w}} \mathbb{E}_i[\mathcal{L}(y_i, F(\mathbf{x}_i; \mathbf{w}))]. \quad (2.1)$$

Gradient descent is an optimization algorithm where the parameters are estimated in successive increments as follows,

$$\mathbf{g}_m = \frac{\partial}{\partial \mathbf{w}_m} \left( \mathbb{E}_i[\mathcal{L}(y_i, F(\mathbf{x}_i; \mathbf{w}_m))] \right), \quad \triangleright \text{gradient} \quad (2.2)$$

$$\mathbf{p}_{m+1} = -\rho_m \mathbf{g}_m, \quad \triangleright \text{parameter update} \quad (2.3)$$

$$\mathbf{w}_m = \sum_{j=0}^m \mathbf{p}_j, \quad \triangleright \text{parameter estimate} \quad (2.4)$$

where  $\rho_m$  is the learning rate at iteration  $m$  and  $\mathbf{p}_0$  is a random initialization of the parameters.

## Functional Gradient Boosting

Functional gradient boosting (Friedman, 2001), commonly known as gradient boosting, is performing gradient descent in function space. That is, instead of computing parameter updates in successive iterations, functional gradient boosting estimates the function. The functional gradient-boosting algorithm starts with an initial model ( $\psi_0$ ) and then iteratively adds the model to estimate the function ( $F = \psi_0 + \psi_1 + \dots + \psi_m$ ). The objective of functional gradient boosting is to estimate the function that best fits the dataset,

$$\arg \min_F \mathbb{E}_i[\mathcal{L}(y_i, F(\mathbf{x}_i))].$$

In functional gradient descent the functions are estimated in successive increments as follows,

$$\tilde{y}_{mi} = \frac{\partial}{\partial F_m} \left( \mathcal{L}(y_i, F(\mathbf{x}_i)) \right), \forall i = 1 \dots N \quad \triangleright \text{gradient} \quad (2.5)$$

$$\psi_{m+1} = -\rho_m \arg \min_{\psi} \mathbb{E}_i[\mathcal{L}(\tilde{y}_{mi}, \psi(\mathbf{x}_i))], \quad \triangleright \text{function update} \quad (2.6)$$

$$F_m = \sum_{j=0}^m \psi_j. \quad \triangleright \text{function estimate} \quad (2.7)$$

where  $\rho_m$  is the learning rate at iteration  $m$  (often estimated with line search) and  $\psi_0$  is an randomly initialized model. Essentially, the functional gradient boosting algorithm calculates functional gradients for each example ( $\tilde{y}_{mi}$ ) in iteration  $m$ . The regression dataset  $(\{\mathbf{x}_i, \tilde{y}_{mi}\}_{i=1}^N)$  is generated and a model  $\psi_m$  is fitted on this dataset.  $\psi_m$  is then added to the final model ( $F$ ). This is repeated till convergence or some fixed number of iterations.

A common optimization objective for regression tasks is to minimize an  $L_2$  loss function, also known as squared error loss. A common objective function used for binary classification tasks is binomial deviance (BD), also known as logistic loss.  $L_2$  loss and BD loss are defined as follows,

$$L_2(y, F(\mathbf{x})) = (y - F(\mathbf{x}))^2 \quad (2.8)$$

$$BD(y, F(\mathbf{x})) = \log(1 + e^{-2yF(\mathbf{x})}) \quad (2.9)$$

The functional gradient ( $\tilde{y}$ ) for the  $L_2$  loss is  $y - F(\mathbf{x})$  and for the BD loss is  $\frac{y}{(1 + \exp(2yF(\mathbf{x})))}$ .

## Functional Gradient Boosted Trees

When each model  $\psi_m$  of the functional gradient boosting is a decision tree (CART) (Breiman et al., 1984), it is called gradient-boosted trees, also referred to as gradient-boosting machines (GBM). Annual survey on *State of Machine Learning and Data Science* by Kaggle has consistently found gradient-boosting trees to be the most popular ML algorithm among complex methods (Kaggle, 2020, 2021, 2022). One of the reasons for its popularity is the availability of user-friendly, highly optimized, and open-source implementations of gradient-boosted trees in Python. Popular ones include Scikit-learn (Pedregosa et al., 2011), LightGBM (Ke et al., 2017), CatBoost (Dorogush et al., 2017), and XGBoost (Chen and Guestrin, 2016).

### 2.3 Qualitative influence information

Qualitative influence (QI) between two variables describes the direction of their relationship. A QI statement indicates how a change in one variable correlates with the other variable.



Figure 2.2: Illustration of monotonic influences between two random variables  $X$  and  $Y$ . (a) Isotonic influence (b) Antitonic influence

Including the qualitative information in machine learning models has shown a significant advantage in performance and speed up conversion for sparse and noisy datasets (Altendorf et al., 2005; Yang and Natarajan, 2013). We leverage the QI information to improve the performance of gradient-boosted trees in Chapter 3. This section required background for the two major types of qualitative influences: monotonic and synergistic.

### 2.3.1 Monotonic influence

Monotonic influence describes a direct relationship between two variables. For example, a monotonic influence statement “As BMI increases, neck circumference increases” indicates that the probability of greater neck circumference increases with an increase in BMI.

**Definition 1.** A random variable  $X$  has a **monotonic influence** on a random variable  $Y$  if higher values of  $X$  stochastically result in higher (or lower) values of  $Y$ . It is denoted by  $X \overset{M+}{\prec} Y$  (or  $X \overset{M-}{\prec} Y$ ).

The monotonic influence relation  $X \overset{M+}{\prec} Y$  that indicates the higher value of  $X$  stochastically results in a higher value of  $Y$  is called isotonic influence. Figure 2.2a illustrates an isotonic relationship between random variables  $X$  and  $Y$ . The monotonic influence relation  $X \overset{M-}{\prec} Y$  that indicates the higher value of  $X$  stochastically results in a lower value of  $Y$  is called antitonic influence. Figure 2.2b illustrates an antitonic relationship between random variables  $X$  and  $Y$ .

### 2.3.2 Synergistic influence

A synergistic influence statement describes the interaction between the influences. Two variables synergistically influence the third variable if their joint influence is greater than their separate, statistically independent influences. Synergic influence can capture influences like “An increase in BMI increases the risk of high blood pressure in patients with a family history of hypertension more than patients without a family history.”

**Definition 2.** *Two random variables  $A$  and  $B$  have a **synergistic influence** (SI) on variable  $Y$ , denoted by  $A, B_{<}^{S+}Y$ , if increasing the value of  $A$  has a greater effect on  $Y$  for a higher value of  $B$  than the lower value of  $B$ . Both  $A$  and  $B$  should necessarily have the same monotonic relationship with  $Y$ .*

Similarly, a *sub-synergistic influence* (sub-SI), denoted by  $A, B_{<}^{S-}Y$ , indicates that while  $A$  and  $B$  have increasing monotonic influence on  $Y$ , the joint influence is lesser than their separate, statistically independent influence.

### 2.3.3 QI for decision-making

While it is difficult for human experts to provide faithful probabilities, it is easier to provide qualitative knowledge about influences between variables (Helsper et al., 2004). In most real-world problems such as healthcare, finance, marketing, social science, etc., human experts have a considerable amount of knowledge about QIs. This knowledge could potentially improve the performance of the ML models in the presence of data sparsity, class imbalance, and/or high dimensionality. Traced back to 1990 (Wellman, 1990), the QI between random variables has a long history of being applied to decision-making models. Researchers have studied the use of QIs in a broad variety of application areas, such as finance (Kim and Han, 2003; Chen and Li, 2014), housing (Potharst and Feelders, 2002), medical research (Yet et al., 2014), computer vision problems (de Campos et al., 2008; Tong and Ji, 2008), etc.

Specifically, monotonic influences are applied to a wide range of ML models, from Support Vector Machines (Bartley et al., 2016a) to Deep Lattice Networks (You et al., 2017). Cano et al. (2019) surveys ML algorithms and their performance under monotonicity constraints for the classification tasks.

## **2.4 Sequential decision making**

A fraction of decision-making problems can be formulated as classification or regression tasks. However, a significant fraction of problems require successive decisions and are better formulated as a problem of finding optimal behavior. Such multi-step sequential decision-making problems often require reasoning over the long-term utility and impact of a decision and the uncertainty over future situations. Sequential decision-making problems consist of a dynamic system that is capable of producing different states when certain actions are performed. Such dynamic systems are mathematically formulated in two different ways, a decision process or a transition system. These two formulations make different assumptions and have been a major focus of research by two different subfields of AI. The subfield reinforcement learning (Sutton and Barto, 1998) focuses on stochastic, nondeterministic, partially observable, utility-based dynamic systems formulated as Markov Decision Processes (MDPs). The subfield of automated planning (Ghallab et al., 2004) focuses on deterministic, fully observable, goal-based transition systems formulated as state-transition systems. This section provides the required background for Chapters 4–7 that focuses on sequential decision-making.

### **2.4.1 Reinforcement learning**

The field of Reinforcement Learning (RL) concerns itself with the problem of how an autonomous agent can interact with the environment and learn the desired behavior from experience. As shown in Figure 2.3, an RL framework consists of an autonomous agent and an environment. The agent observes the state of the environment, performs an action,

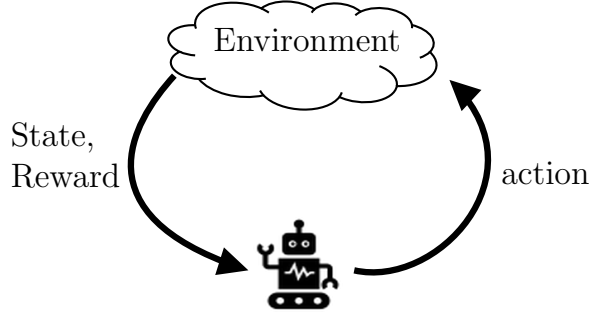


Figure 2.3: Illustration of an RL framework.

observes the reward and the next state, performs the next action, and so on. The goal of an agent is to interact with the environment and learn an optimal policy that maximizes the observed reward. At its core, the agent has to trade off between exploring the environment and exploiting the existing knowledge to maximize the reward. This problem of learning by interaction is framed as MDPs in RL.

**Definition 3.** A *Markov decision process* (MDP) is defined as a tuple  $\langle S, A, T, R, \gamma \rangle$ , with a set of states  $S$ , a set of actions  $A$ , a transition function  $T : S \times A \times S \rightarrow [0, 1]$ , a reward function  $R : S \times A \times S \rightarrow \mathbb{R}$ , and a discount factor  $\gamma$ .

The solution to the MDP is an optimal policy  $\pi^* : S \times A \rightarrow [0, 1]$  that yields the maximum utility value for all the states  $s \in S$ . While there can be different ways of defining the utility value (for example, average reward, total reward, etc), in this dissertation we define the utility value,  $V^\pi(s)$ , of a state  $s$  as the expected discounted cumulative reward received upon following the policy  $\pi$  starting from state  $s$ .

$$V^\pi(s_t) = \mathbb{E}_{(T, \pi)} \left[ \sum_{i=0}^{\infty} \gamma^i r_{t+i}^\pi \right] \quad (2.10)$$

where  $r_t^\pi = R(s_t, a_t^\pi, s_{t+1})$  is the reward observed upon following the policy and expectation is over the state transition probability distribution and the probability of the action in the policy. A Q-value of a state and action pair,  $Q^\pi(s, a)$ , is defined as the expected cumulative

discounted reward received upon starting from state  $s$ , taking action  $a$ , and following the policy  $\pi$  thereafter.

$$Q^\pi(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma V^\pi(s_{t+1}) \quad (2.11)$$

An optimal policy  $\pi^*$  yields the optimal value,  $V^*(s)$ , and the optimal Q-value,  $Q^*(s, a)$ . So, an action in a given state is optimal if it yields an optimal Q-value. Hence, many RL algorithms focus on computing the optimal value or Q-value and use it to determine the optimal policy. The computation of the Q-value is difficult when the transition function  $T$  and the reward  $R$  are not known. In the absence of these functions, approximation algorithms are proposed that iteratively update the value function or the Q-value function. Q-learning by Watkins (1989) is one such iterative algorithm that we used in our experiments.

## Q-Learning

The idea behind the powerful Q-learning algorithm is the following Bellman optimality constraint that defines the optimal Q-value at a state with respect to the optimal Q-value of the neighboring states,

$$Q^*(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma \sum_{s_{t+1}} T(s_t, a_t, s_{t+1}) \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}). \quad (2.12)$$

This recursive definition of Q-value provides the following iterative Q-update equation,

$$\hat{Q}_{k+1}(s_t, a_t) \leftarrow (1 - \alpha) \hat{Q}_k(s_t, a_t) + \alpha \left( r(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} \hat{Q}_k(s_{t+1}, a_{t+1}) \right), \quad (2.13)$$

where  $\alpha$  is the learning rate.

The Q-learning algorithm starts with a random initialization of Q-values for each state-action pair. An agent first observes state  $s$  of the environment, performs an exploratory action  $a$ , observes the rewards and the next state, and updates the Q-value  $Q(s, a)$  using the above equation. This process is repeated till convergence or max number of steps is reached in an environment. The intuition behind this iterative algorithm is that if each state-action

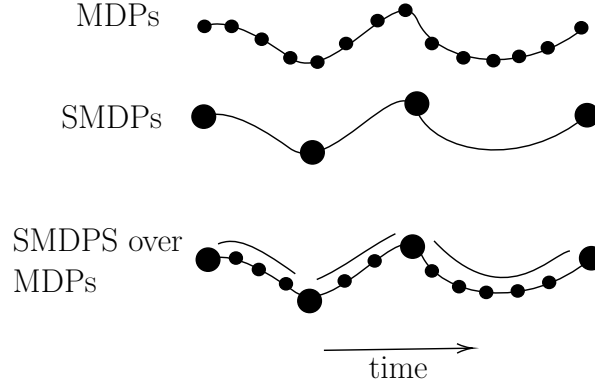


Figure 2.4: Comparison of actions in MDPs, SMDPs, and HRL

pair is visited often enough, then the updates would converge towards optimal Q-function. Recently, many neural variants of Q-learning were proposed for complex RL environments, including Deep Q-Network (DQN) (Mnih et al., 2013, 2015), Double DQN (van Hasselt et al., 2016), Dueling DQN (Wang et al., 2016), Noisy DQN (Fortunato et al., 2018), Rainbow DQN (Hessel et al., 2018) etc.

## Hierarchical RL

While RL is generally successful, it suffers from the curse of dimensionality when the action space is too large and/or state space is infeasible to enumerate. Humans simplify complex problems by abstracting away details and decomposing actions into hierarchies. Several researchers have proposed to model the **temporal-abstraction** in RL by composing a hierarchy over the action space (Dietterich, 1998; Sutton et al., 1998; Parr and Russell, 1998). By modeling actions as hierarchies, researchers extended the primitive action space by adding temporally extended actions. These approaches are broadly categorized as hierarchical RL (HRL) approaches.

MDPs conceive time as a discrete step. A chosen action in an MDP only persists for a single step. Semi-MDPs (SMDPs) allow temporally extended actions of varying length over a continuous time, as represented in the first two trajectories of Figure 2.4. By extending the

primitive action space of the MDP by adding temporally extended actions, HRL approaches superimpose MDPs and SMDPs as shown in the last trajectory of Figure 2.4. By leveraging the action hierarchies, HRL methods improve the exploration of the environment. They are efficient in learning due to their ability to decompose larger tasks into smaller sub-tasks. They are amenable to transfer and generalization (Dietterich, 1998). They have shown significant benefits in problems with long horizons and sparse rewards. Different HRL approaches include the Options framework (Sutton et al., 1998), the MAXQ framework (Dietterich, 1998), hierarchical abstract machines (HAM) (Parr and Russell, 1998), etc. Among these, we use the options framework in Chapters 4–6.

In our work, we use two HRL approaches, the options framework and the hierarchical DQN. The options framework (Sutton et al., 1998) models temporally extended actions as *options*. An option  $p = \langle I_p, \pi_p, \beta_p \rangle$  consists of three components: a set of states where the option can be initiated  $I_p$ , an option policy  $\pi_p$ , and a termination condition  $\beta_p : S \rightarrow [0, 1]$  that describes the probability of option termination. A policy is learned for each option. At each step, the agent either decides to take a primitive (single-step) action or initiate a temporally extended option. When an option is initiated, the agent uses the respective option policy to decide on actions until the option can be terminated. Kulkarni et al. (2016) builds upon the options framework and proposes a hierarchical DQN (h-DQN). HDQN is a two-level hierarchical RL framework where a meta-controller chooses the goals and a controller chooses the actions. An internal critic provides an intrinsic reward to the controller for the chosen goal. The controller maximizes the intrinsic reward and the meta-controller maximizes the extrinsic, environmental reward.

## Relational RL

Many structured domains have objects with different properties and relations between these objects. Relational RL (RRL) focuses on RL in such structured domains. RRL approaches

learn higher-order rules for acting in a domain and are better at generalizing across similar objects (Dzeroski et al., 2001; Tadepalli et al., 2004). In RRL, the MDP definition has been extended to the relational MDPs (RMDPs) (Fern et al., 2006). In Chapters ?? we focus on the relational MDPs. But before introducing them we must make a distinction between the state predicates and the action predicates. State predicates indicate properties or relations between the entities of the world, whereas action predicates indicate an activity, movement, or step. For example, in a taxi domain (see Fig. 6.1a) predicate `east/0` is an action predicate and the atom `east()` moves the taxi in the east direction. For simplicity, we assume the action predicates have zero arity.

**Definition 4.** *Given a first-order language  $\mathcal{L}$  consisting of a set of constants  $C$ , a set of state predicates  $P$ , and a set of action predicates  $Y$ , a **relational MDP** (RMDP) is defined as a tuple  $\langle S, A, T, R, \gamma \rangle$ , where:*

- *$S$  is a finite set of states defined over all the ground atoms generated by constants  $C$  and state predicates  $P$*
- *$A$  is a finite set of actions defined as a set of ground atoms generated by constants  $C$  and action predicates  $Y$*
- *$T : S \times A \times S \rightarrow [0, 1]$  is a probabilistic transition function*
- *$R : S \times A \times S \rightarrow \mathbb{R}$  is a reward function*
- *$\gamma \in [0, 1)$  is the discount factor*

RMDPs can either be converted to propositional MDPs and solved using standard RL approaches, or they can be solved using RRL approaches. RRL approaches include symbolic as well as neural approaches. Notable work using symbolic methods include relational TILDE trees with Q-learning (Dzeroski et al., 2001), RRL-TG that replaces TILDE with

incremental tree learning algorithm (Driessens et al., 2001), approximate policy iteration (API) with decision-lists (Fern et al., 2006), relational gradient-boosted Q-learning (GBQL) and relational boosted fitted Q-learning (RBFQ) (Das et al., 2020). Neural RRL approaches include neural logic machines (NLM) (Dong et al., 2019) that showed the ability to solve blocks world with up to 50 blocks, neural logic reinforcement learner (NLRL) (Jiang and Luo, 2019), off-policy differentiable logic RL (OPDLRL) (Zhang et al., 2021) that uses differentiable ILP ( $\partial$ ILP) (Evans and Grefenstette, 2018), the work by Kimura et al. (2021) that used Logical Neural Network (LNN) (Riegel et al., 2020) for text-based RL games, and various deep relational RL works that use graph-based neural networks (GNNs) architectures with off-policy learning algorithms (Li et al., 2020; Zambaldi et al., 2019; Janisch et al., 2021).

### 2.4.2 Planning

Planning is the model-based approach to sequential decision making (Ghallab et al., 2004; Geffner and Bonet, 2013). A planning agent uses a transition model of the domain to compute how the states would evolve upon taking certain actions and selects the action that achieves the goal. The description of the transition model is called a planning domain. A planning domain  $\mathbf{D} = \langle \mathcal{L}, \mathbf{O} \rangle$  consists of a first-order language  $\mathcal{L}$  and a finite set of schematic operators  $\mathbf{O}$ . A schematic operator is defined as  $c = \langle h(c), \text{pre}(c), \text{eff}^+(c), \text{eff}^-(c) \rangle$ , consisting of a lifted atom,  $h(c)$ , often referred to as the head; a first-order formula called preconditions  $\text{pre}(c)$ ; and two disjoint sets of atoms,  $\text{eff}^+(c)$  and  $\text{eff}^-(c)$ , describing the positive (add) and negative (delete) effects of executing the operator. All terms that appear in the literals in  $\text{pre}$ ,  $\text{eff}^+$ , and  $\text{eff}^-$  also appear as arguments of  $h(c)$ . Hence, a schematic operator can be grounded by substituting the head atom,  $h(c)\theta$ . A ground operator  $c\theta$  can be applied in a state  $s$  if a substitution  $\theta$  satisfies the precondition  $\text{pre}(c)$  at  $s$ , i.e.  $s \models \text{pre}(c)\theta$ . On applying the ground operator  $c\theta$ ,  $s$  transitions to another state  $s' = (s \setminus \text{eff}^-(c)\theta) \cup \text{eff}^+(c)\theta$ .

A classical planning task is defined as a tuple  $\Pi = \langle \mathbf{D}, s_0, g \rangle$ , where  $s_0$  is an initial state and  $g$  is a goal specified as a conjunction of literals. A plan for the planning task is a sequence of ground operators, which when executed in a state  $s$  results in a state satisfying  $g$ . A language and syntax for planning domains called Planning Domain Definition Language (PDDL) was released for the 1998 AI Planning Systems Competition (McDermott, 2000). PDDL has since been revised and extended to various versions (Ghallab et al., 1998; Fox and Long, 2002, 2003; Edelkamp and Hoffmann, 2004; Gerevini and Long, 2005) and has become a standard input syntax for most of the existing planners.

## Hierarchical Planning

Solving complex planning tasks with large state space can be very compute-intensive. To reduce the computational cost, the powerful idea of hierarchical problem solving is adapted in planning, which is referred to as hierarchical planning. In hierarchical planning, the goal is decomposed into sub-goals and multiple plans are generated at different abstraction hierarchies. Essentially, hierarchical planners first identify an abstract plan for the planning task and then refine each abstract action in the abstract plan to find a concrete plan. Hierarchical planning is most suitable in domains that satisfy the downward refinement assumption (Bacchus and Yang, 1991), where it is assumed that, given a concrete action plan exists, every abstract plan can be refined to a concrete action plan without backtracking across the abstract plan. Hierarchical Task Networks (HTN) (Erol et al., 1994; Nau et al., 1999) is one such hierarchical planning approach.

HTN consists of two types of tasks: *primitive* and *compound*. A *primitive task* is defined using preconditions and effects (equivalent to operators in classical planning). A primitive task is a single-step action that can only be executed in a state which satisfies the task preconditions. Upon executing a primitive task the propositions of the state change as dictated by the task effects. A *compound task* is a temporally extended action that requires one or

```

Methods:
    <t: transport(G) # PASSENGER IN TAXI
    pre:  $\exists X, (at\_dest(X) \in G)$ 
     $\tau$ : {drop(X), transport(G)} >

    <t: transport(G) # NOT AT DESTINATION
    pre:  $\exists X, at\_dest(X) \in G \wedge \neg at\_dest(X)$ 
     $\tau$ : {pickup(X), transport(G)} >

Operators:
    < h: pickup(X)
    pre:  $\exists L, at(X, L) \wedge \neg in\_taxi(X)$ 
    eff: taxi-at(L)  $\wedge$  in-taxi(X)
     $\beta$ : in-taxi(X) >

    < h: drop(X)
    pre:  $\exists L, in\_taxi(X) \wedge dest(X, L),$ 
    eff: taxi-at(L)  $\wedge$  at-dest(X)  $\wedge \neg in\_taxi(X)$ 
     $\beta$ : at-dest(X) >

Initial State (s): {at(p1,r),
                    taxi-at(l3), dest(p1,d1),
                     $\neg at\_dest(p1), \neg in\_taxi(p1)$ }

Goal condition (g): {at-dest(p1)}

```

Figure 2.5: Example of a hierarchical planning task consisting of methods, operators, initial state, and goal condition.

more primitive tasks to be executed in a partially ordered manner. One or more methods are defined for each compound task. A method is described as a three tuple, consisting of compound tasks, preconditions, and a sequence of partially ordered tasks (primitive or compound). When a state satisfies the method precondition, the compound task can be decomposed in that state to the sequence of partially ordered tasks. The problem of achieving a goal condition, starting from a given initial state, is also posed as a compound task (or set of compound tasks). Each compound task is recursively decomposed using methods, till a satisfying sequence of primitive tasks is identified. A sequence of primitive tasks is considered satisfying if executing that sequence starting at the initial state results in a state that satisfies the goal condition.

A hierarchical planning domain  $\mathbf{D} = \langle \mathcal{L}, \mathbf{O}, \mathbf{M} \rangle$  is similar to a classical planning domain with an additional set of methods  $\mathbf{M}$ . A method is a triple  $m = \langle t(m), \text{pre}(m), \tau(m) \rangle$ , where  $t$  is a compound-task,  $\text{pre}$  is a precondition for the method, and  $\tau$  is an ordered sequence of compound-tasks or operators. A task  $t(m)$  can be decomposed into a sequence of sub-tasks  $\tau(m)$  at a state  $s$  if a substitution  $\theta$  satisfies the preconditions in  $s$ , i.e.  $s \models \text{pre}(m)\theta$ . A hierarchical planner solves the planning task by recursively decomposing the goal into a sequence of sub-tasks. Figure 2.5 provides an example of a hierarchical planning task.

PDDL does not support compound tasks and methods. Höller et al. (2020) proposed a *Hierarchical Domain Description Language* (HDDL) as an extension of the STRIPS fragment of PDDL2.1 defined in Fox and Long (2003). Both PDDL and HDDL are first-order languages and use first-order formulas over predicates for goal description.

## 2.5 Statistical relational AI

Statistical Relational AI (Raedt et al., 2016), also referred to as StarAI, is a field of AI that deals with learning and reasoning under uncertainty in relational domains. StarAI is at the intersection of the following three paradigms, as shown in Figure 2.6.

1. Logic, the field of logical representations deals with reasoning for relational data.
2. Probability, statistical approach to handle uncertainty, noise, and missing values.
3. Learning, the field of machine learning use data-driven approaches to learn model structures and parameters.

Logical representations like propositional and predicate logic are useful to reason about conjunction, negation, and implications. They capture rich relational structures of the domain and allow powerful generalization across different objects, but they are not great tools for reasoning about uncertainty. Probability, on the other hand, is a great tool to reason about uncertainty, beliefs, and possible worlds. The powerful idea of using structured

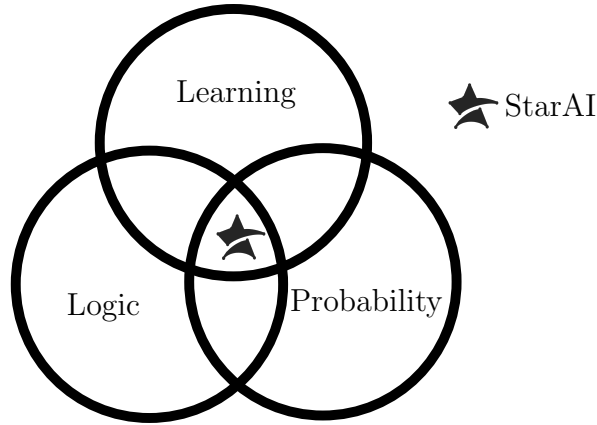


Figure 2.6: Statistical Relational AI (StarAI) combines Logic, Probability, and Learning.

graphical representations for the computation of probability and uncertainty gave rise to the vast literature on probabilistic graphical models (Koller and Friedman, 2009), including Bayesian networks, Markov networks, dependency networks, etc. Both the logic and graphical model research have intersections with Machine learning. For instance, inductive logic programming (?) is a research area at the intersection of computational logic and machine learning.

Eventually, the ideas from logic, machine learning, and probabilistic graphical models started emerging together for complex domains with rich relational structures. Various models were proposed at the intersection of logical representations and probabilistic graphical models, for instance, probabilistic relational models (PRMs) (Friedman et al., 1999), Bayesian logic programs (BLPs) (Kersting and Raedt, 2001), relational Markov networks (Taskar et al., 2002), relational dependency networks (RDNs) (Neville and Jensen, 2004), Markov logic networks (MLNs) (Richardson and Domingos, 2006) etc. These multitudes of models and formalisms were collected under the umbrella of Statistical Relational Learning (SRL) (Getoor and Tasker, 2007). The field of SRL focuses on the representation, inference, and learning of these relational probabilistic graphical models. Motivated by the capability of SRL approaches, researchers began looking at the use cases of SRL in various

AI problems and coined the term StarAI. The field of StarAI was born with the motivation to

“explore the minimal perturbations required for each of the AI subfields to start using statistical relational (SR) techniques” (Star-AI (2010))

### 2.5.1 First-order conditional influence language

Like probabilistic graphical models, SRL models also describe influences in the domains. However, instead of specifying the influence between random variables or objects, SRL models specify the influence between attributes of classes. For instance, probabilistic graphical models allow describing influences like “Harsha’s grades depend on Harsha’s intelligence”, but SRL models allows specification of influences like “For all students, their grades depend on their intelligence”. The exact syntax of modeling such influence might differ for each SRL approach. In Chapters 4–6, we use one of the SRL languages—First-order Conditional Influence (FOCI) language. FOCI was first introduced in Natarajan et al. (2005) and described in Natarajan and Altendorf (2005). Going beyond the classical SRL approaches like PRMs and BLPs, FOCI allows the modeling of qualitative influences in the domain (see § 2.3 for qualitative influence).

FOCI language consists of statements of the form,

$$\text{IF } \langle \text{condition} \rangle \text{ THEN } \langle \text{influent} \rangle \text{ QINF } \langle \text{resultant} \rangle, \quad (2.14)$$

where **condition** and **influent** are a finite set of literals, **resultant** is a single literal, and QINF indicates the qualitative influence. A FOCI statement encodes the qualitative influence (QINF) of **influent** on the **resultant**. Below is an example of a FOCI statement encoding the following QI: “If a student takes a course, then the grade in that course is monotonically influenced by the student’s intelligence.”

$$\begin{aligned} \text{IF } \text{student}(S), \text{ course}(C), \text{ register}(R, S, C) \\ \text{THEN } \text{intelligence}(S, I) \text{ QINF } \text{grade}(R, G), \end{aligned} \quad (2.15)$$

```

student(amy), student(bella), student(conner),
intelligence(amy,ia), intelligence(bella, ib), intelligence(conner, ic),
course(stats), course(math), course(economics),
register(r1, amy, stats), student(r2, conner, math)
grade(r1, g1), grade(r2, g2)

```

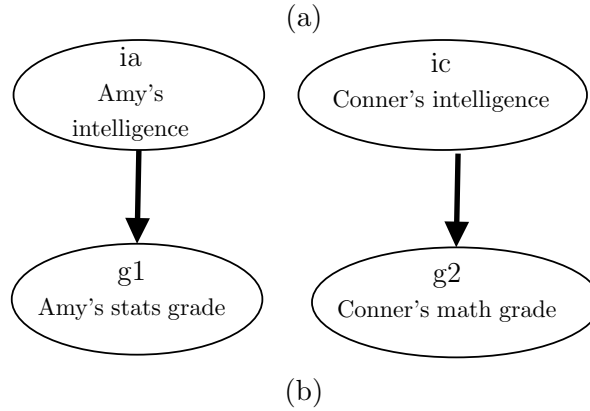


Figure 2.7: (a) An instance of a domain with all the objects. (b) Grounded Bayesian network obtained for the FOCI statement in Equation 2.15

Each FOCI statement can be associated with the conditional probability function that describes the probability distribution of the resultant conditioned on the influents. Given a particular instantiation of the domain, FOCI statements produce a ground Bayesian network. Figure 2.7 demonstrates a ground Bayesian network obtained for the above FOCI statement.

## 2.6 Neurosymbolic AI

While neural computational and learning approaches have made significant advances in various pattern recognition and generation problems, they are shown to be limited in reasoning capabilities. This has led the research community to combine symbolic reasoning capabilities with neural learning approaches, giving rise to a class of approaches called *Neurosymbolic AI* (d'Avila Garcez et al., 2015). In his 2020 Robert S. Engelmore Memorial Lecture, Kautz

(2022) proposed a taxonomy for neurosymbolic AI; surveying the various ways of combining the neural and symbolic advancements from related fields of deep learning (Goodfellow et al., 2016), probabilistic graphical models (Koller and Friedman, 2009), graph neural networks (Battaglia et al., 2018), statistical relational learning (Raedt et al., 2016), etc. This taxonomy provides a good overview of the field of Neurosymbolic AI. The taxonomy includes the following 6 architectures.

### **Type 1: Symbolic Neuro Symbolic**

The first category is **Symbolic Neuro Symbolic**. Here a symbolic entity is encoded as a vector embedding and passed to a neural network model. The output from the neural network is a vector that is decoded back to symbolic representation, as shown in Figure 2.8a. This is perhaps the most common approach seen in natural language processing for language translation or question-answering tasks.

### **Type 2: Symbolic[Neuro]**

The second category, **Symbolic[Neuro]**, represents the hybrid systems where a symbolic solver employs one or more neural subroutines, illustrated in Figure 2.8b. AlphaGo (Silver et al., 2017) is a fine illustration of this category, with Monte Carlo Tree Search using a neural network as a subroutine for heuristic estimation.

### **Type 3: Neuro | Symbolic**

The third category is for systems where symbolic and neural models are co-routines and are leveraged for different tasks in a big pipeline. In **Neuro | Symbolic**, neural and symbolic routines communicate with each other either to extract information or to improve individual as well as collective performance of the system, see Figure 2.8c. DeepProbLog (Manhaeve et al., 2018) and NeurASP (Yang et al., 2020) are illustrative examples of this category. Both

of these framework trains a neural network to learn probabilistic atoms, which are used in symbolic reasoning, and the loss from reasoning is used for computing gradients.

#### **Type 4: Neuro: Symbolic $\rightarrow$ Neuro**

**Neuro: Symbolic  $\rightarrow$  Neuro** is the fourth category. Here, symbolic knowledge is used to either modify the training procedure of the Neural Network or provide an architecture or initial weights to bias the neural network. This type is illustrated in Figure 2.8d. Knowledge-based Artificial Neural Networks (KBANN) (Towell and Shavlik, 1994) is a great exemplar of this category. It uses symbolic rules to define a neural network and uses backpropagation to refine the rules. Some of the recent works in this category involve Lifted Restricted Boltzmann Machines (LRBM) (Kaur et al., 2020), TensorLog (Cohen et al., 2020), and Neuro-symbolic forward reasoner (Shindo et al., 2021).

#### **Type 5: Neuro\_{Symbolic}**

The fifth category is **Neuro\_{Symbolic}** where there is a tight coupling of symbolic data and the neural operations. Symbols are converted to tensors for neural processing as shown in Figure 2.8e. However, neural processing is tightly coupled with symbolic operations and manipulations. Logic Tensor Network (Serafini et al., 2017) and Neural Logic Machines (Dong et al., 2019) fall under this category.

#### **Type 6: Neuro[Symbolic]**

Finally, the last category is **Neuro[Symbolic]**, transpose of Type 2 (see 2.8f). Here, overall Neural model performs symbolic reasoning by either learning the relations between the symbols or paying attention to selected symbols at a certain point. Graph Neural Networks, with message passing among neighbors and attention to selected relations, fall in this category (Lamb et al., 2020).

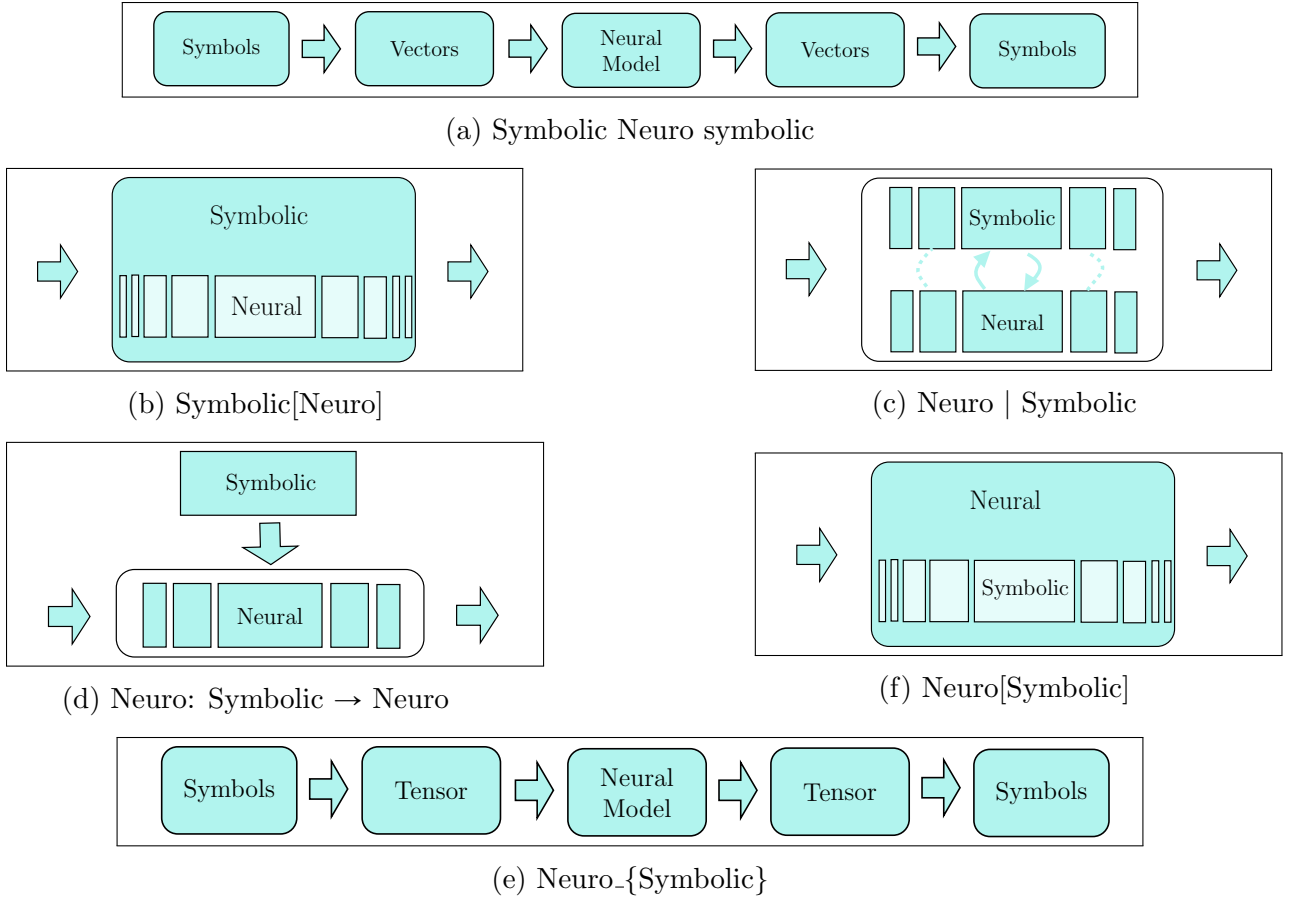


Figure 2.8: Taxonomy of Neurosymbolic approaches.

## **PART I**

### **SPARSE AND NOISY DOMAINS**

## CHAPTER 3

### INCORPORATING QUALITATIVE INFLUENCE INFORMATION

Tree-based gradient boosting methods (see §2.2) are very powerful and are successfully used in various challenging problems (Olson et al., 2018). These methods learn the ensemble of trees to fit the training data instances as closely as possible. Hence, while the prediction is optimized for training data, it is unreliable in the region of the sample space where data is scarce or not available, as well as in the region where the majority of the data is noisy. Rich qualitative constraints such as monotonic and synergistic influences (see §2.3) can provide additional information to improve prediction in the sparse and noisy region of the sample space. In this chapter, we introduce an approach to incorporate rich qualitative domain knowledge while learning tree-based gradient-boosting methods. We present the Knowledge-intensive Gradient Boosting framework (Kokel et al., 2020) that can effectively and efficiently incorporate qualitative influence information and improve the performance of the tree-based gradient boosting models. KiGB uses human guidance as soft constraints to gently nudge the leaf values to follow the qualitative influence statements.

### 3.1 Introduction

In many real-world domains such as healthcare and logistics, qualitative influence statements such as monotonicities and synergies are quite natural and easy to obtain. For instance, a physician could easily explain that “as the A1C number increases, the risk of heart attack increases”; a domain expert in logistics could explain that “as the distance between the source and destination increases, the price of shipping increases”. Ignoring such valuable information while learning a model appears wasteful.

Several prior works exist on incorporating such domain knowledge in the context of learning ML models, for classification task (Cano et al., 2019) and for regression tasks such as

isotonic regression (Robertson et al., 1988). Including such constraints can yield significantly faster and better convergence compared to using only data, specifically in probabilistic models like Bayesian network (Altendorf et al., 2005; Yang and Natarajan, 2013). The key advantage of employing such constraints appears to be in the cases of noisy and sparse domains.

Inspired by these successes, we propose a novel method to adapt the qualitative constraints in the successful gradient-boosted trees framework<sup>1</sup>. We develop a unified approach that works for both classification and regression tasks. Specifically, we use the qualitative influence between the feature and target variables, obtained from the domain expert, as a soft constraint while learning the gradient-boosted trees. Our hypothesis, which we evaluate empirically, is that model learned by using the qualitative influence would show significant benefit over the model learned only from the noisy data. Especially when the data is scarce.

In this chapter, we make the following important contributions: (1) we derive the first unified framework for gradient-boosting that can be adapted to classification and regression settings. (2) we show how to use the constraints in the gradient updates to directly modify the model parameters, instead of altering the data distribution. (3) Inspired by the use of knowledge for learning SVMs (Fung et al., 2002), we provide an interpretation of the resulting framework using margins. (4) Finally, and most importantly, we demonstrate both the effectiveness and efficiency of the proposed approach in multiple domains—15 standard benchmark domains for classification and regression tasks and 2 real data sets (including a novel logistics data set).

The code and data used in this chapter are available for public use at the following URL: <https://github.com/starling-lab/KiGB>. The rest of the chapter is organized as follows. Section 3.2 reviews related work on using qualitative constraints in trees. Then Section 3.3 presents our unified algorithm for both the classification and regression settings.

---

<sup>1</sup>Why gradient boosted trees? Refer to Appendix A.1

It also explains the margin interpretation of the KiGB constraints. Section 3.4 presents the empirical evaluation of the proposed algorithm on several classifications and regression tasks. Finally, Section 3.5 concludes by summarizing the key insights and contributions and outlines some future research directions.

## 3.2 Related work

A variety of approaches have been proposed to incorporate monotonic influence in trees. Ben-David (1995) modified splitting criteria to consider both the entropy and the order-ambiguity score. Makino et al. (1996); Potharst and Bioch (1999) append new corner-elements to the dataset to learn monotonic function. Feelders and Pardoel (2003); Bioch and Popova (2002) prune the non-monotonic branches after learning the tree. Bioch and Popova (2002) proposed to relabel the dataset and remove all non-monotonic instances. van de Kamp et al. (2009) adjusted the probability values at the leaf nodes in case of a violation by using isotonic regression functions. González et al. (2015); Bonakdarpour et al. (2018) adapt the above approaches to random-forest decision tree ensembles. Bartley et al. (2016b) leverages the formulation of the random-forest decision tree ensemble as a weighted neighborhood function and proposes a re-weighting scheme subject to monotonicity constraints. While effective, none of these approaches have been successfully adapted to gradient boosting.

Two approaches proposed for boosted trees include González et al. (2016) and Bartley et al. (2019). González et al. (2016) proposed a pruning mechanism for monotone AdaBoost. It first learns the whole tree, compares each branch split with all other splits to establish monotonicity, and prunes the non-monotonic branches. Hence, it is inefficient. Another approach, *Monoensemble* by Bartley et al. (2019), converts each tree to monotone rules and then re-calculates the leaf values (coefficients) to ensure monotonicity. They propose two methods for coefficient recalculation: Logistic regression and Naive Bayesian

techniques. Monoensemble is proposed for the random forest ensemble, however, the implementation extends the approach to gradient boosting for classification tasks. Random forest Monoensemble outperforms all the previous approaches for classification tasks and also guarantees global monotonicity. To this effect, we compare our proposed approach with Monoensemble (MONO).

LightGBM (LGBM) (Ke et al., 2017) and XGBoost (Chen and Guestrin, 2016) provide an option to learn monotonic gradient-boosted trees. Both these libraries constrain the node splits while performing a greedy search to learn a monotonic tree. At each node of the tree, after selecting the splitting variable, the *mean* of the left and the right subtree is used as the bounding constraint for future splits. The leaf values of the future split in the left subtree are upper bounded ( $\leq$ ) by the *mean* and the leaf values of the right subtree are lower bounded ( $>$ ) by the *mean*. This approach is very restrictive (as shown by Bartley et al. (2019)), and can overfit the training data. It can be effective for problems that require strict monotonicities and have clean data, without any noise. In our evaluations, we also compare our proposed approach against standard gradient-boosting (LGBM) as well as gradient-boosting with monotonicity constraints (LMC) implemented in the LightGBM library.

### 3.3 Knowledge-intensive gradient boosting

KiGB leverages qualitative influences, provided by domain experts, to improve learning with functional gradient boosting. Our focus is not to achieve *strict monotonicity*, rather we aim to use the monotonic influences as advice to learn a *loosely monotonic function* that yields faster and better convergence when the data is scarce and noisy. Previous approaches (Monoensemble (Bartley et al., 2019) and LMC (Ke et al., 2017), for example) incorporate monotonic influences as hard constraints in the target function. Instead, KiGB naturally incorporates the influences at each iteration during boosting as soft constraints.

The motivation of this chapter can be illustrated with the following example. Consider a typical case in the logistics domain where even though the monotonic advice like “as the distance between the source and destination increases, the price of shipping increases” holds in general, a trucking company might charge a lower price for some particular long-distance shipping. Various reasons could lead to such scenarios, e.g. driver returning to the home base, convenient parking, next scheduled pick-up, etc. So we do not aim to learn a strictly monotonic function. Rather, we propose an approach to learning a loosely monotonic function that allows the trade-off between the data and the advice. Our approach does not guarantee monotonicity but as shown by the experimental results, for most of the datasets, it has clear benefits over approaches that guarantee monotonicity.

### 3.3.1 Monotonic constraint

As we aim to use monotonic influence for classification and regression tasks, we only consider the monotonic influence between a feature and a target variable. Given an increasing monotonic (isotonic) influence between a feature variable  $A$  and target variable  $Y$  ( $A \stackrel{Q^+}{\prec} Y$ ), Monotonicity definition (see Def. 1) indicates the following order restriction in the expectation,

$$A = a_1 \leq A = a_2 \implies \mathbb{E}[Y \mid A = a_1, \mathbf{C} = \mathbf{c}] \leq \mathbb{E}[Y \mid A = a_2, \mathbf{C} = \mathbf{c}]. \quad (3.1)$$

The above equation indicates that given  $A = a_1$  is less than  $A = a_2$  the expected value of  $Y$  for samples with  $A = a_1$  is less than  $A = a_2$  in the same context, that is when all the other assignments to the other random variables ( $\mathbf{C}$ ) is same. The context encodes the *ceteris paribus*<sup>2</sup> condition.

We extend this order restriction to trees. Consider a tree  $\psi_t$  that splits at node  $n$  with variable  $A$ . The expected values of the left subtree of  $n$  should be no more than the expected

---

<sup>2</sup>*Ceteris paribus* is a Latin phrase that roughly translates to “all else being equal”.

value of the right subtree of  $n$ .

$$\mathbb{E}[\psi(\mathbf{n}_L)] \leq \mathbb{E}[\psi(\mathbf{n}_R)] \quad (3.2)$$

where  $\mathbf{n}_L$  (resp.  $\mathbf{n}_R$ ) is the set of all examples assigned to the left (resp. right) subtree at  $n$ . All the variables that appear in any of the ancestors of the node  $n$  can be seen as the context. KiGB uses this expectation as a constraint on the leaf values and incorporates it into the objective function. The *ceteris paribus* condition is relaxed in this constraint. Instead, an assumption is made that the variables appearing in the ancestor of the node  $n$  are sufficient context.

Inspired by the work of Fung et al. (2002), we incorporate an  $\varepsilon$ -margin in the constraint as follows,

$$\mathbb{E}[\psi_t(\mathbf{n}_L)] - \mathbb{E}[\psi_t(\mathbf{n}_R)] \leq \varepsilon \quad (3.3)$$

The  $\varepsilon$ -margin in the constraint ensures that the monotonicity is robustly enforced up to a (potentially user-specified) tolerance of  $\varepsilon$ . We define a slack variable  $\zeta_n$  to measure the violation of the  $\varepsilon$ -margin monotonicity constraints at node  $n$ ,

$$\zeta_n = \mathbb{E}[\psi_t(\mathbf{n}_L)] - \mathbb{E}[\psi_t(\mathbf{n}_R)] - \varepsilon \quad (3.4)$$

We modify the standard objective of squared-error loss to include a penalty ( $\zeta_n^2$ ) when the advice constraint is violated ( $\zeta_n > 0$ ). So the objective function is defined as,

$$\underset{\psi_t}{\operatorname{argmin}} \underbrace{\sum_{i=1}^N (\tilde{y}_i - \psi_t(\mathbf{x}_i))^2}_{\text{loss function w.r.t data}} + \underbrace{\frac{\lambda}{2} \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \max(\zeta_n \cdot |\zeta_n|, 0)}_{\text{loss function w.r.t. advice}} \quad (3.5)$$

where  $\mathcal{N}(\mathbf{X}_c)$  is the set of all *non-leaf* nodes that split on the monotonic features ( $\mathbf{X}_c$ ) influencing the target variable and parameter  $\lambda$  expresses the relative importance of the advice constraint in the problem. The loss function w.r.t. the advice is a form of hinge loss and is activated only on violation of the advice constraint ( $\zeta_n > 0$ ).

On taking derivation of the modified objective function, we get the following leaf update equation,

$$\psi_t^\ell = \underbrace{\frac{1}{|\ell|} \sum_{i=1}^N \tilde{y}_i \cdot \mathbb{I}(\mathbf{x}_i \in \ell)}_{\text{mean}} + \underbrace{\frac{\lambda}{2} \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \mathbb{I}(\zeta_n > 0) \zeta_n \cdot \left( \frac{\mathbb{I}(\ell \in \mathbf{n}_R)}{|\mathbf{n}_R|} - \frac{\mathbb{I}(\ell \in \mathbf{n}_L)}{|\mathbf{n}_L|} \right)}_{\text{penalty for advice violation}} \quad (3.6)$$

where,  $\mathbb{I}(\ell \in \mathbf{n}_R)$  represents whether leaf  $\ell$  belongs to the right subtree of node  $n$  and  $|\ell|$  represents number of examples at the leaf node  $\ell$ .

### Derivation of the update equation

The leaf update equation (Equation 3.6) is derived from the modified objective function (Equation 3.5) as follows.

Let us represent each tree as a sum of its leaves (j).

$$\psi_t(\mathbf{x}) = \sum_{j \in \psi_t} \psi_t^j \cdot \mathbb{I}(\mathbf{x} \in j)$$

where,  $\mathbb{I}(\mathbf{x} \in j)$  represents whether example  $\mathbf{x}$  is captured in leaf  $j$ . Using this in the modified objective function (Equation 3.5), we obtain the following equation,

$$\sum_{i=1}^N \left( \tilde{y}_i - \sum_{j \in \psi_t} \psi_t^j \cdot \mathbb{I}(\mathbf{x}_i \in j) \right)^2 + \frac{\lambda}{2} \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \max(\zeta_n \cdot |\zeta_n|, 0)$$

We take the partial derivative of this objective with respect to the model parameters (leaf values).

$$\sum_{i=1}^N \frac{\partial}{\partial \psi_t^\ell} \left( \tilde{y}_i - \sum_{j \in \psi_t} \psi_t^j \cdot \mathbb{I}(\mathbf{x}_i \in j) \right)^2 + \frac{\lambda}{2} \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \frac{\partial}{\partial \psi_t^\ell} (\max(\zeta_n \cdot |\zeta_n|, 0))$$

The derivation of  $\max(\zeta_n \cdot |\zeta_n|, 0)$  is,

$$\frac{\partial}{\partial \psi_t^\ell} \max(\zeta_n \cdot |\zeta_n|, 0) = \begin{cases} \frac{\partial}{\partial \psi_t^\ell} (\zeta_n)^2, & \text{if } \mathbb{I}(\zeta_n > 0) \\ 0, & \text{otherwise} \end{cases}$$

Plugging this in the previous equation we get,

$$\begin{aligned} & -2 \sum_{i=1}^N \left( \tilde{y}_i - \sum_{j \in \psi_t} \psi_t^j \cdot \mathbb{I}(\mathbf{x}_i \in j) \right) \cdot \mathbb{I}(\mathbf{x}_i \in \ell) + \frac{\lambda}{2} \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \mathbb{I}(\zeta_n > 0) \frac{\partial}{\partial \psi_t^\ell} (\zeta_n)^2, \\ & -2 \sum_{i=1}^N \left( \tilde{y}_i - \sum_{j \in \psi_t} \psi_t^j \cdot \mathbb{I}(\mathbf{x}_i \in j) \right) \cdot \mathbb{I}(\mathbf{x}_i \in \ell) + \frac{\lambda}{2} \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \mathbb{I}(\zeta_n > 0) 2\zeta_n \frac{\partial}{\partial \psi_t^\ell} (\zeta_n). \end{aligned}$$

Substituting  $\zeta_n$ ,

$$-2 \sum_{i=1}^N (\tilde{y}_i - \psi_t^\ell) \cdot \mathbb{I}(\mathbf{x}_i \in \ell) + \lambda \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \mathbb{I}(\zeta_n > 0) \cdot \zeta_n \frac{\partial}{\partial \psi_t^\ell} (\mathbb{E}[\boldsymbol{\psi}_t(\mathbf{n}_L)] - \mathbb{E}[\boldsymbol{\psi}_t(\mathbf{n}_R)] - \varepsilon)$$

Substituting  $\mathbb{E}[\boldsymbol{\psi}_t(\mathbf{n})]$ ,

$$\begin{aligned} & -2 \sum_{i=1}^N (\tilde{y}_i \cdot \mathbb{I}(\mathbf{x}_i \in \ell)) + 2\psi_t^\ell \sum_{i=1}^N \mathbb{I}(\mathbf{x}_i \in \ell) + \\ & \lambda \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \mathbb{I}(\zeta_n > 0) \cdot \zeta_n \left( \frac{\partial}{\partial \psi_t^\ell} \left( \frac{1}{|\mathbf{n}_L|} \sum_{\mathbf{x}_i \in \mathbf{n}_L} \psi_t(\mathbf{x}_i) \right) - \right. \\ & \left. \frac{\partial}{\partial \psi_t^\ell} \left( \frac{1}{|\mathbf{n}_R|} \sum_{\mathbf{x}_i \in \mathbf{n}_R} \psi_t(\mathbf{x}_i) \right) \right) \end{aligned}$$

Here,  $\sum_{i=1}^N \mathbb{I}(\mathbf{x}_i \in \ell)$  is the number of samples at leaf node  $\ell$ , we use the notation  $|\ell|$  for it.

$$\begin{aligned} & -2 \sum_{i=1}^N (\tilde{y}_i \cdot \mathbb{I}(\mathbf{x}_i \in \ell)) + 2\psi_t^\ell \cdot |\ell| + \\ & \lambda \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \mathbb{I}(\zeta_n > 0) \cdot \zeta_n \left( \frac{1}{|\mathbf{n}_L|} \sum_{\mathbf{x}_i \in \mathbf{n}_L} \mathbb{I}(\mathbf{x}_i \in \ell) - \frac{1}{|\mathbf{n}_R|} \sum_{\mathbf{x}_i \in \mathbf{n}_R} \mathbb{I}(\mathbf{x}_i \in \ell) \right) \end{aligned}$$

$\sum_{\mathbf{x}_i \in \mathbf{n}_L}^N \mathbb{I}(\mathbf{x}_i \in \ell)$  is true only if the  $\ell \in \mathbf{n}_L$  and it is equal to  $|\ell|$ .

$$- 2 \sum_{i=1}^N (\tilde{y}_i \cdot \mathbb{I}(\mathbf{x}_i \in j)) + 2\psi_t^\ell \cdot |\ell| + \lambda \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \mathbb{I}(\zeta_n > 0) \cdot \zeta_n \left( \frac{\mathbb{I}(\ell \in \mathbf{n}_L) \cdot |\ell|}{|\mathbf{n}_L|} - \frac{\mathbb{I}(j \in \mathbf{n}_R) \cdot |\ell|}{|\mathbf{n}_R|} \right)$$

Now, equating the derivation with zero will give us the following equation for leaf values:

$$\psi_t^\ell = \underbrace{\frac{1}{|\ell|} \sum_{i=1}^N \tilde{y}_i \cdot \mathbb{I}(\mathbf{x}_i \in \ell)}_{\text{mean}} + \underbrace{\frac{\lambda}{2} \sum_{n \in \mathcal{N}(\mathbf{X}_c)} \mathbb{I}(\zeta_n > 0) \zeta_n \cdot \left( \frac{\mathbb{I}(\ell \in \mathbf{n}_R)}{|\mathbf{n}_R|} - \frac{\mathbb{I}(\ell \in \mathbf{n}_L)}{|\mathbf{n}_L|} \right)}_{\text{penalty for advice violation}}$$

where,  $\mathbb{I}(\ell \in \mathbf{n}_R)$  represents whether leaf  $\ell$  belongs to the right subtree of node  $n$  and  $|\ell|$  represents number of examples at the leaf node  $\ell$ .

### 3.3.2 Interpretation of the update equation

Intuitively, if the advice constraint is violated for the node  $n$  then the penalty applies a total correction of  $\lambda \cdot \zeta_n$  on all the leaves in the subtrees. *It is worth noting that the penalty applied to each subtree is inversely proportional to the number of examples in that subtree.* So, when there is significant data available, the advice is scaled down. The tree is first constructed by evaluating the splitting variable w.r.t the standard squared-error loss and then the leaf values are updated as per the modified objective.

High values of  $\lambda$  force aggressive advice-based updates when a constraint is violated and the influence of data is reduced. Alternately, small values of  $\lambda$  make the advice constraint less strict, and the trees depend more on data. When  $\lambda = 0$ , the objective is a standard tree learning that relies only on the data. Negative values of  $\varepsilon$  enforce strict margins while positive values allow overlapping margins. In extreme cases, to enforce the expected value of the left subtree to be strictly less than the expected value of the right subtree by some  $k$ , i.e.  $\mathbb{E}[\psi_t(\mathbf{n}_L)] + k \leq \mathbb{E}[\psi_t(\mathbf{n}_R)]$ , we set  $\varepsilon = -k$ . In other cases, to allow margin of  $k$  for

violation, i.e.  $\mathbb{E}[\psi_t(\mathbf{n}_L)] \leq \mathbb{E}[\psi_t(\mathbf{n}_R)] + k$ , we set  $\varepsilon = k$ . These interpretations of  $\lambda$  and  $\varepsilon$  are also evident in the experiments with hyperparameters.

### 3.3.3 Equilibrium between advice and data

Given the different degrees of the plausibility of expert advice and levels of noise in the data, an ideal approach should allow for different extents to which the monotonic constraints can influence  $\psi$ . This section illustrates the significance of achieving such an equilibrium between the experts' knowledge and data. Consider the noisy dataset shown in Figure 3.1a with feature  $A$  on the horizontal axis,  $B$  on the vertical axis, and different colors representing different regression values of the target  $Y$ . Assume that an expert provided the *monotonic influence advice*  $A \stackrel{Q^+}{\prec} Y$  for this data. The noisy data clearly violates this constraint in region R1 & R2. Different use cases may require treating this anomaly as conflict or noise. In some cases, this anomaly might be an important conflicting pattern in the data that should be captured by the model, while in others, advice is more significant and the anomaly may be overlooked as noisy observations. The latter case is especially when it is known that some sensors collecting the data are not sensitive enough or have failed.

Figure 3.1b illustrates the decision boundaries of a gradient-boosted model (with 2 trees) learned with a vanilla gradient-boosting algorithm that does not incorporate the monotonic constraints. In this case, as can be seen, the model can possibly become incorrect due to two specific reasons—missing data (as in region R5) or noisy data (as in R1). Figure 3.1c illustrates the decision boundaries learned by an LMC model with split-constrained boosted trees. LMC overfits the training data by finding an unnatural split that best optimizes the objective. This phenomenon is further elaborated in Section 3.3.4.

Figure 3.2 illustrate the decision boundaries learned by the KiGB approach that uses monotonic constraints. As the  $\lambda$  values indicate the relative importance given to the advice, upon increasing  $\lambda$  the regression value in the regions on the right (R2 and R3) are gradually

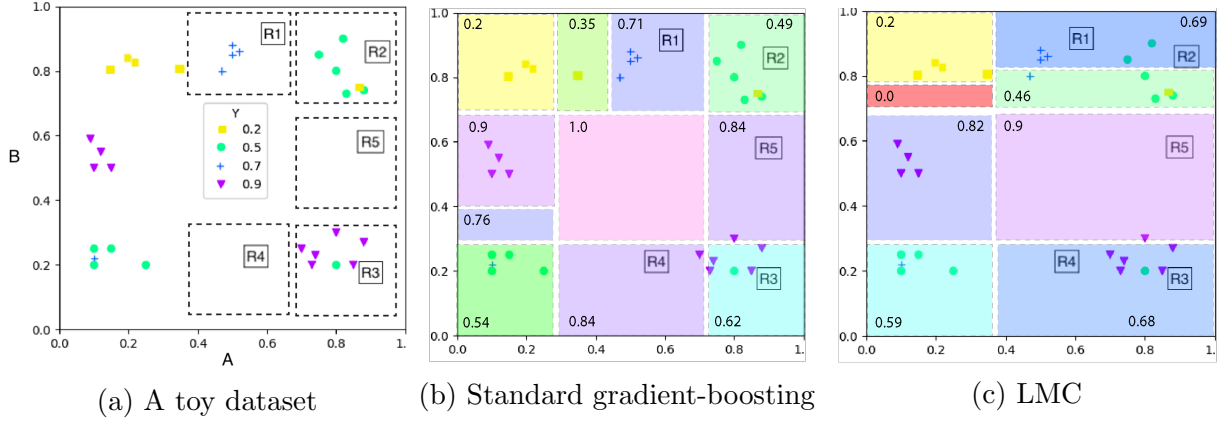


Figure 3.1: An example to illustrate the need for equilibrium between data and advice. (a) A toy dataset. The horizontal axis represents the feature variable  $A$ , the vertical axis represents the feature variable  $B$ , and markers represent the target variable  $Y$ . An expert provides a monotonic influence statement  $A \stackrel{Q}{\prec} Y$ . Region R1 and R2 violate the advice. (b) Illustration of the decision boundary learned by standard gradient-boosting approach. (c) Illustration of decision boundary learned by monotonic gradient-boosting approach.

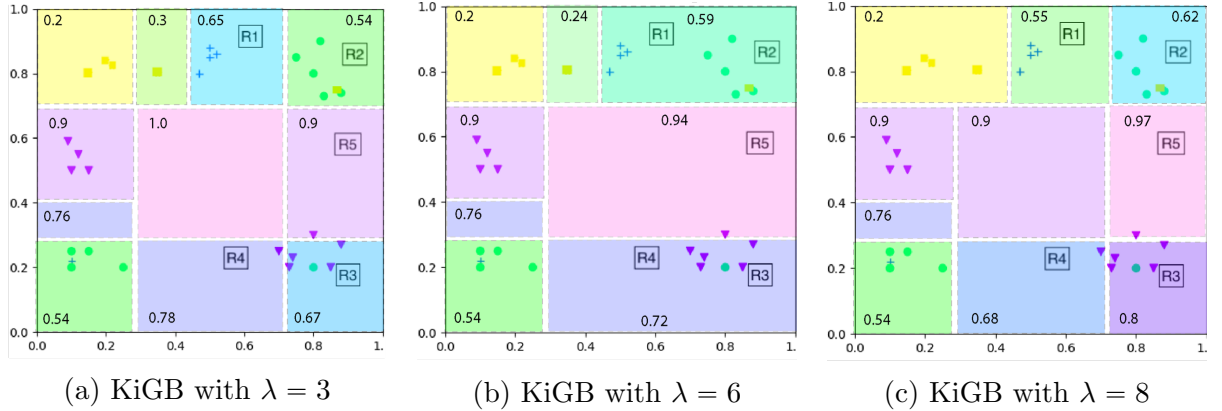


Figure 3.2: Illustration of the decision boundaries learned by KiGB approach with different relative importance to advice (different  $\lambda$  values).

increased and the regression value in the left regions (R1 and R4) are decreased. This shows that as the importance of the advice increases ( $\lambda$  increases), the function learned is increasingly positive monotonic w.r.t.  $A$ , i.e. biased by advice. We specifically see the advantage of advice for the predicted regression value in the region R5, where *no training examples* are available.

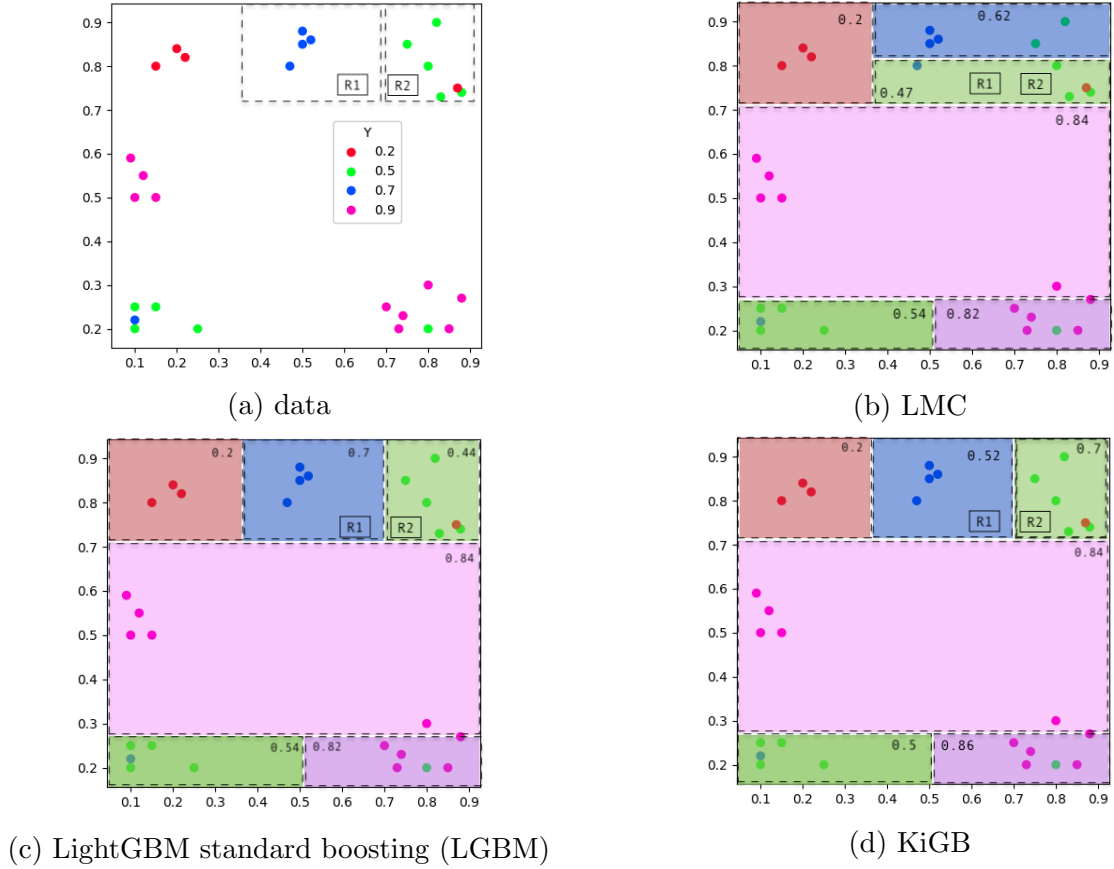


Figure 3.3: Illustration of the overfitting by LMC. As can be seen, LGBM, without any monotonic influence statements, learned an incorrect model due to the presence of noisy data. With LMC, the model learns a monotonic function but it overfits the training data. LKiGB provides a correction to the LGBM and generalizes to a better model.

### 3.3.4 Overfitting by strict monotonicity

When a model is fitted w.r.t data under strict monotonicity constraints, as done by LMC, it may overfit the data. We illustrate this with an example in this section. Consider the noisy data shown in Figure 3.3a, with feature  $A$  on the horizontal axis,  $B$  on the vertical axis, and different colors representing different regression values of the target  $Y$ . Assume that some expert provided the monotonic influence advice between  $A$  and  $Y$ — $A \prec_{<}^+ Y$ —for this data.

The noisy data clearly violates this constraint in regions R1 & R2. In the scenario where the advice is significantly more important than the noisy data, it might seem reasonable to

use the strict monotonic boosting provided by LightGBM (LMC). However, as seen in Figure 3.3b the LMC overfits the training data by splitting horizontally in the region R1 & R2. The standard boosting method (LGBM) (Figure 3.3c), on the other hand, uses natural splits but has no way of correcting the noise. Our LKiGB approach (Figure 3.3d) uses the monotonic influence information from the expert to provide correction and learn a monotonic function.

### 3.3.5 KiGB algorithm

We propose to learn a KigB model in two stages. That is we first fit a tree to the dataset and then adjust the leaf values by adding appropriate penalties. Algorithm 1 presents the complete procedure for learning a KiGB model for a regression task. Algorithm 1 starts with a mean value as an initial estimate in Line 2 for optimizing the mean-squared error (Equation 2.8) and iteratively adds a model fitted to data and advice. In line 4 standard functional gradient is computed. In Line 5, a tree is fit to the computed gradient w.r.t. the data, accounting for the mean from Equation 3.6. Then, for each leaf of the regression tree, the penalty term from Equation 3.6 is evaluated (w.r.t. the monotonic features  $\mathbf{X}_c$ , relative

---

#### Algorithm 1 Knowledge-intensive Gradient Boosting

---

INPUT: Data  $(\mathbf{x}, y)$ , # trees  $M$ , monotonic features  $\mathbf{X}_c$ ,  $\lambda$ ,  $\varepsilon$

OUTPUT:  $\psi(\mathbf{x})$

```

1: function KiGB( $\mathbf{x}, y, M, \mathbf{X}_c, \lambda, \varepsilon$ )
2:    $\psi(\mathbf{x}) = \psi_0(\mathbf{x}) = \text{mean}(y)$ 
3:   for  $m = 1$  to  $M$  do
4:      $\tilde{y} = y - \psi(\mathbf{x})$  ▷ Compute gradient
5:      $\psi_m(\mathbf{x}) = \text{tree}(\tilde{y}, \mathbf{x})$  ▷ Learn the next tree
6:     for  $\ell$  in  $\psi_m$  do
7:        $\psi_m^\ell(\mathbf{x}) = \psi_m^\ell(\mathbf{x}) + \text{penalty}^\ell(\mathbf{X}_c, \lambda, \varepsilon)$  ▷ From equation 3.6
8:     end for
9:      $\psi(\mathbf{x}) = \psi(\mathbf{x}) + \psi_m(\mathbf{x})$  ▷ Update the function
10:  end for
11:  return  $\psi(\mathbf{x})$ 
12: end function

```

---

importance  $\lambda$ , and margin  $\varepsilon$ ) and applied in Line 7. Finally, the tree is added to the current model (Line 9). The updated leaf values ( $\psi_m^\ell$ ) guide the gradients ( $\tilde{y}$ ) in the next iteration and help achieve faster convergence. Note that while the algorithm takes the number of trees  $M$  as input, it is easy to use any convergence criteria such as a change in the likelihood to create the ensembles. Thus, the non-parametric property of the gradient-boosting algorithm can still be preserved.

### 3.3.6 Classification

The modified KiGB objective function, Equation 3.5, can be easily adapted to any other loss function like deviance or exponential-loss to extend it for classification. Our experiments for the classification tasks used mean-squared error loss to fit the tree and binomial deviance (BD, Equation 2.9) for the functional gradient. So, our classification experiments follow the same process as shown in Algorithm 1 with modification of lines 2 and 4. The initial estimate ( $\psi_0$ ) is the `median`( $y$ ) and gradients ( $\tilde{y}$ ) are computed for BD as mentioned in Section 2.2.

### 3.3.7 Extensions

Since the KiGB modifies the tree learning objective function w.r.t the qualitative constraint and not the functional gradient objective, it is easy to use this approach on any tree-based learning methods like a decision tree, random forests, AdaBoost, relational regression trees, etc. Additionally, Odom and Natarajan (2018) shows that there exists a close connection between qualitative constraints and preferences. In specific cases, preferences can be reduced to qualitative constraints. Hence, the KiGB framework can be leveraged with preferences as well. Consider the example advice shown in Odom and Natarajan (2018), *if any car passes an agent on the right, then the agent should move into the right lane*. This advice is represented as a preference rule  $r = \langle F, l+, l- \rangle$  with preferred label  $l+ = \text{move right}$  and avoid label  $l- = \text{stay}$ . This can be converted to monotonic influence as  $F \stackrel{Q^+}{\prec} l+$  and

$F^Q_{<}l-$ . Once converted, the framework can be directly applied while learning the model. Theoretically analyzing the convergence properties of our framework, on the other hand, remains an interesting future direction.

### 3.4 Experiments

Our evaluations explicitly aim to answer the following questions:

- Q1.** Can KiGB effectively utilize monotonic knowledge?
- Q2.** How does KiGB compare against previous boosting with the monotonicity approach for classification?
- Q3.** How does KiGB compare against a monotonic ensemble method for regression?
- Q4.** How sensitive are the learned models to the KiGB hyperparameters?
- Q5.** How effective is KiGB on real data (potentially noisy)?
- Q6.** Does the use of advice benefit when data is scarce?

#### 3.4.1 Datasets

We perform thorough evaluations of KiGB over 15 standard datasets. All of these standard datasets were obtained from the UCI Machine Learning repository<sup>3</sup> (Dua and Graff, 2017), except the following: Boston and California housing datasets were obtained from StatLib datasets archives<sup>4</sup> (Vlachos and Meyer, 2005) and Windsor housing dataset were obtained

---

<sup>3</sup>UCI ML repository: <https://archive.ics.uci.edu/ml/index.php>

<sup>4</sup>StatLib datasets archives: <http://lib.stat.cmu.edu/datasets/>

Table 3.1: Datasets used in the experiments. The first 5 datasets have binary classification tasks, the next 10 have regression tasks and the last two are real-world datasets described in Section 3.4.5. The second column either refers to the literature from where we got the monotonic features and/or lists the feature names used for experiments. Features in **bold** have negative influence ( $X \stackrel{Q}{\prec} Y$ ) and others have positive influence ( $X \stackrel{Q}{\succ} Y$ ).

Dataset	Monotonic Features ( $\mathbf{X}_c$ )
Classification datasets	
Adult	You et al. (2017)
Australian	Duivesteijn and Feelders (2008)
Car	Bartley et al. (2016b)
Cleveland	Bartley et al. (2016b)
Ljubljana	Bartley et al. (2016b) + age
Regression datasets	
Abalone	Length,Diameter, Height, Shell weight
Autompg	Cano et al. (2019)
Autoprice	horsepower, peak-rpm, city-mpg, highway-mpg
Boston	RM, <b>CRIM</b> , <b>PTRATIO</b>
California	Total Rooms, Total Bedrooms
CPU	Cano et al. (2019)
Crime	population, racepctblack, <b>racepctWhite</b> , agePct65up, pctWPubAsst, <b>PctKids2Par</b> , <b>PctKids2Par</b> , <b>PctYoungKids2Par</b>
Redwine	<b>volatile acidity</b> , citric acid, sulphates, alcohol
Whitewine	<b>volatile acidity</b> , citric acid, sulphates,alcohol
Windsor	lot
Realworld datasets	
Logistics	miles, team driver, holiday, new year, average fuel price
HELOC	FICO (2018)

from JAE Data Archive<sup>5</sup> (Anglin and Gencay, 1996). We utilize qualitative constraints discussed in previous literature when available. Table 3.1 overviews the datasets and the monotonic constraints used in our evaluations for each dataset.

Across all of the domains, the test sets for the experiments were created by randomly selecting 20% of the available data. Five iterations were performed by sampling 80% of the remaining data as the training set. The same training and test sets were used across different methods. These splits are made available publicly on the project Github.<sup>6</sup> We fixed the number of tree estimators to 30 for our experiments and report all the results with a learning rate of 0.1.

### 3.4.2 Standard gradient boosting baselines

First, we compare the gradient boosting implementation of Scikit-learn (SGB) (Pedregosa et al., 2011) against our KiGB framework implemented in Scikit-learn (SKiGB). The results in Table 3.2 show that for 4 out of 5 classification datasets and for 6 out of 10 regression datasets SKiGB yields significantly better performance. In other domains, the performance of SKiGB is comparable to SGB. Values in **bold font** indicate statistical significance at  $p\text{-value} = 0.1$ . Thus, we affirmatively answer Q1—using monotonic influences while learning has added advantage over learning only from the data in the majority of domains, for both the tasks of regression and classification.

### 3.4.3 Monotonic gradient boosting baselines

Next, we compare our KiGB framework against two approaches that incorporate monotonic constraints for boosting: Monoensemble (MONO) and LightGBM monotonic constraints

---

<sup>5</sup>JAE Data Archive, Anglin and Gencay (1996): <http://qed.econ.queensu.ca/jae/1996-v11.6/anglin-gencay/>

<sup>6</sup>Training and test splits of the 10 standard dataset: <https://github.com/starling-lab/KiGB/tree/master/datasets>

Table 3.2: Standard baselines: Comparison of performance of SKiGB and SGB. The performance measure used is accuracy for classification tasks (the higher the better) and mean squared error for regression tasks (the lower the better).

Dataset	SKiGB	SGB	Dataset	SKiGB	SGB
Classification tasks					
Adult	<b>0.855</b>	0.853	Cleveland	<b>0.737</b>	0.677
Australian	<b>0.855</b>	0.83	Ljubljana	<b>0.696</b>	0.621
Car	0.984	0.982			
Regression tasks					
Abalone	<b>5.377</b>	5.491	CPU	<b>0.185</b>	0.204
Autompg	<b>9.793</b>	13.623	Crime	2.211	2.296
Autoprice	8.866	8.945	Redwine	<b>0.381</b>	0.419
Boston	24.065	21.493	Whitewine	<b>0.426</b>	0.439
California	47.159	47.468	Windsor	<b>3.9</b>	4.626

Table 3.3: Monotonic baselines: Comparison of accuracy of KiGB and monotonic boosting approaches for classification tasks.

Dataset	SKiGB	MONO	LKiGB	LMC
Classification tasks				
Adult	0.855	0.857	<b>0.865</b>	0.863
Australian	0.855	<b>0.884</b>	<b>0.878</b>	0.867
Car	<b>0.984</b>	0.765	<b>0.971</b>	0.959
Cleveland	0.737	0.74	<b>0.757</b>	0.73
Ljubljana	<b>0.696</b>	0.611	0.721	0.718

Table 3.4: Monotonic baselines: Comparison of mean squared error of LKiGB and LMC for regression tasks.

Dataset	LKiGB	LMC	Dataset	LKiGB	LMC
Regression tasks					
Abalone	4.786	4.797	CPU	<b>0.206</b>	0.208
Autompg	<b>8.047</b>	8.33	Crime	1.834	1.847
Autoprice	<b>14.953</b>	15.614	Redwine	<b>0.382</b>	0.397
Boston	<b>15.496</b>	16.292	Whitewine	<b>0.45</b>	0.467
California	<b>48.517</b>	50.94	Windsor	<b>2.524</b>	2.634

(LMC). Since there is a significant difference in the implementation of gradient-boosted trees in Scikit-learn and LightGBM<sup>7</sup>, we compare MONO which was implemented in the Scikit-learn library with SKiGB and LMC with our KiGB framework implemented in LightGBM (called LKiGB). Bartley et al. (2019) proposed MONO for binary and multi-class classification, so we could only compare MONO with SKiGB for classification datasets. Table 3.3 shows that for 2 of the 5 datasets, SKiGB outperforms MONO significantly and in one dataset, Australian Credit, MONO surpasses SKiGB. LightGBM (Ke et al., 2017), one of the popular libraries for gradient boosting trees, has the ability to define monotonic constraints for regression as well as classification settings. Tables 3.3 and 3.4 compare our LKiGB with LightGBM’s monotonic constraints (LMC) for standard datasets. LKiGB achieves better or comparable performance for most of the datasets. These comparisons with MONO and LMC answer Q2 and Q3 positively.

### 3.4.4 Robustness to the hyperparameters

From our experiments, we find that KiGB achieves consistent performance in the following ranges:  $\varepsilon \in [-1, 1]$  and  $\lambda \in [0, 5]$ , but this range may vary based on the range of the regression value of the target variable. We demonstrate the robustness of the KiGB framework with respect to the hyperparameters on two datasets—Ljubljana and Autotmpg. We picked one classification and one regression dataset, for which KiGB showed significant improvement over both baselines.

Figures 3.4a and 3.4b compare a standard vanilla gradient boosting and a monotonic boosting baseline against KiGB for various values of the hyperparameters. Standard gradient boosting with LightGBM is referred to as LGBM. It is visible from the figures that KiGB provides consistent improvement regardless of the hyperparameters used. These figures are

---

<sup>7</sup>LightGBM additionally uses exclusive feature bundling and gradient-based one-side sampling (Ke et al., 2017)

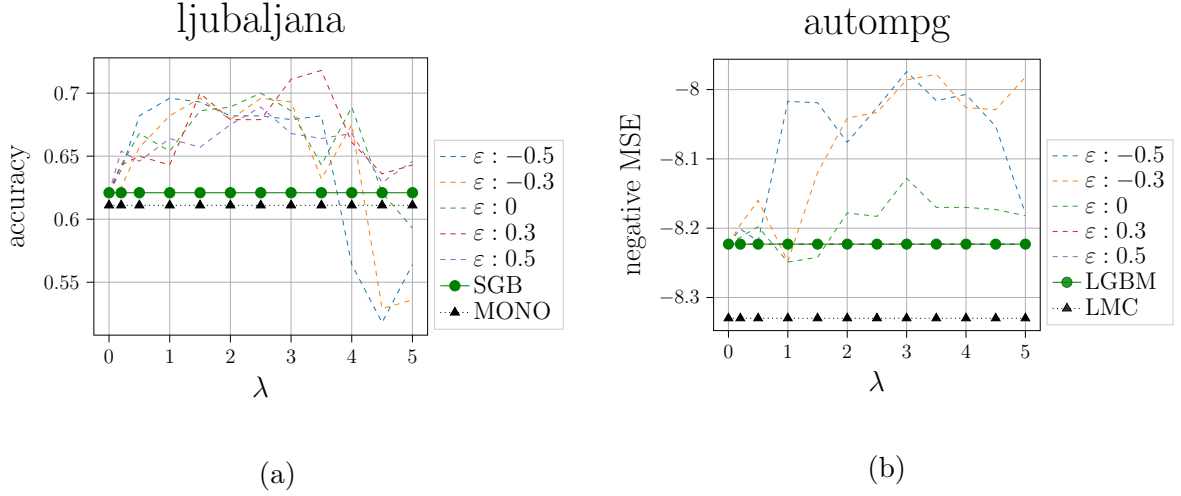


Figure 3.4: Analysis of sensitivity to hyperparameters:  $\lambda$  &  $\varepsilon$ . (a) Comparison of accuracy with SGB and MONO for a classification task. (b) Comparison of negative mean-squared error with LGBM and LMC for a regression task. The higher the better.

representative of the trend we see across datasets. When  $\lambda = 0$ , KiGB is equivalent to vanilla boosting, which relies only on the data. For negative  $\varepsilon$ , lower  $\lambda$  values show improvement but higher  $\lambda$  values reduce the performance. This is in unison with our understanding of reduced performance when we enforce wider margins. For positive  $\varepsilon$ , we see regular performance even for higher  $\lambda$ . For datasets that do not have noise, or there is no violation of the advised constraints, there will be no penalty and hence the performance for positive  $\varepsilon$  will be equivalent to the standard gradient boosting. Autompg is one such dataset and hence the  $\varepsilon = 0.3$  and  $\varepsilon = 0.5$  lines in Figure 3.4b overlap with the LGBM line. With this, we answer Q4. KiGB framework is robust to hyperparameters in noisy datasets and when the dataset is not noisy, it will perform equivalent to the standard gradient boosting for positive  $\varepsilon$ . We recommend using cross-validation to tune these parameters for different problems.

Table 3.5: Comparison of KiGB with standard and monotonic baselines on real-world datasets.

Dataset	LKiGB	LGBM	LMC
Logistics (MSE)	<b>1.851</b>	1.898	1.889
Dataset	SKiGB	SGB	MONO
HELOC (accuracy)	<b>0.717</b>	0.7	0.688

### 3.4.5 Real data sets

To evaluate the utility of KiGB on real-world noisy data we perform experiments on two non-standard data sets: Logistics and HELOC. First, we describe both these data sets and then present the results.

In the Logistics dataset, we consider a regression task of predicting the price of shipping goods by trucks. 859 records for shipments from South Carolina to Florida were collected between June 2017 and May 2018 by a logistics platform, Turvo Inc<sup>8</sup>. These records included 10 different cities and furnished the following information: pickup location, delivery location, distance in miles, pickup and delivery date, average fuel price, load-to-truck ratios, and price of the shipment. We use a subset of these features and some derived features for these experiments. The dataset is made available along with the code. Subject matter experts (SMEs) from Turvo apprised us of general trends in the logistics industry. Specifically, Turvo SMEs provided advice such as market trends dictate that shipping prices will increase (1) around major holidays; (2) when the shipment has to be driven continuously by alternating drivers day and night; (3) when the miles driven by the driver exceeds a certain threshold; (4) finally when the fuel prices have surged; etc. These generalized advice statements that model the domain trends were then converted to monotonic influences between various random variables.

---

<sup>8</sup>Turvo Inc, <https://turvo.com>

The second dataset is the Home Equity Line of Credit (HELOC) applications made by real homeowners, released as part of the FICO explainable machine learning (xML) challenge (FICO, 2018)<sup>9</sup>. The classification task here is to predict whether applicants will repay their HELOC account within 2 years and classify them into bad or good categories. Consumers who have made at least one payment past the due date of 90 days, in 24 months since the credit account was opened were labeled “bad”. Conversely, the consumers who made all the payments within the due date were labeled “good”. FICO also released expected patterns of monotonicity for many feature variables. In our experiments, we use these monotonic constraints to compare KiGB with other models.

Table 3.5 displays the capability of KiGB in comparison with the vanilla gradient boosting and monotonic boosting approaches and answers Q5. For the regression task in the Logistics dataset, we report the mean-squared error and compare LKiGB with LGBM and LMC. For the classification task in the HELOC dataset, we report the accuracy values of SKiGB as compared with SGB and MONO.

### 3.4.6 Learning curve

One major advantage of knowledge-based learning is that it requires fewer training examples as compared to models which learn only from data. We test this hypothesis for the KiGB framework to answer Q6. The learning curve presented in Figure 3.5 compares the performance of KiGB, vanilla gradient boosting, and Monoensemble on the **real-world data set of HELOC**. We sample specified fractions of training data and evaluate all three approaches on the same training/test set. We see that the knowledge-based models (KiGB and MONO) converge to better performance even with fewer training samples while the vanilla gradient boosting (GB) converges gradually with the increasing number of samples.

---

<sup>9</sup>FICO xML challenge, <https://community.fico.com/s/explainable-machine-learning-challenge>.

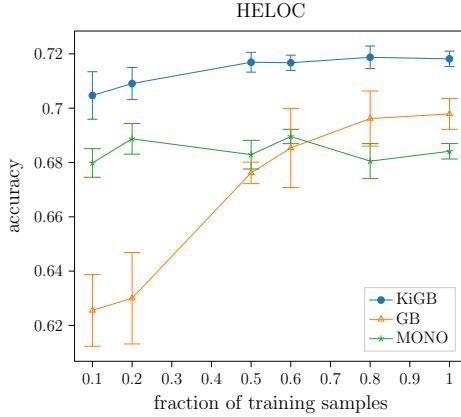


Figure 3.5: Learning curve for classification task in HELOC dataset with KiGB, standard boosting, and monoensemble

Specifically, KiGB is significantly better than just using the data and the default monotonicity method. This clearly shows that when available, qualitative knowledge can not only accelerate learning but could also converge to superior performance.

### 3.5 Summary

We considered the problem of providing rich domain knowledge to the successful gradient-boosting algorithm. Specifically, we presented the use of qualitative constraints such as monotonicities and synergies in learning a robust, generalized model. There are a few important takeaways: (1) KiGB is **better than learning from only data in most of the domains and is never worse**. This clearly shows that even in cases where the knowledge is not well-informed or is imperfect (such as wine datasets), KiGB does not suffer. (2) When the knowledge is indeed relevant as in the case of HELOC, KiGB achieves **a jump start, better slope for learning, and most importantly, a higher asymptote in performance**. This clearly illustrates the need for knowledge injunction in learning systems.

This chapter focused on using the monotonic influence information but the framework and the penalty function can equivalently incorporate the synergistic influence information. Instead of penalizing the violation of expectation for the nodes with monotonic features

( $\mathbf{X}_c$ ), synergy constrain can penalize the violation of the expectation for the branches which have both the features exhibiting synergic influence on the target. Our initial evaluations of synergic constraints on the benchmark datasets did not find the synergic features together in the same branch. So we were not able to evaluate the synergic formulation.

Our KiGB framework is general-purpose, one that can be adapted easily for both classification and regression tasks. Our comprehensive evaluations across several benchmarks and real-world data sets demonstrate the efficiency and effectiveness of the proposed approach. However, the underlying assumption in KiGB is that the monotonic influences in the domain are available from a domain expert. In some cases, this assumption might not hold. So in a follow-up work (described in Chapter 9), we propose an algorithmic approach to identify the qualitative influences.

## **PART II**

### **MULTI-TASK LEARNING AND GENERALIZATION**

## CHAPTER 4

### INTEGRATING RELATIONAL PLANNING AND REINFORCEMENT LEARNING

A major challenge in the field of sequential decision-making (see §2.4) is to generalize to multiple tasks (multi-task learning) and a varying number of objects in the domain. Explicitly modeling the compositionality of a problem structure provides a tremendous advantage in generalization capabilities (Lake et al., 2015; Huang et al., 2019; Li et al., 2019; Devin et al., 2019). Additionally, state abstractions also enable sample-efficient learning and better task transfer in complex environments (Andre and Russell, 2002a; Nitti et al., 2015; Konidaris, 2019). In this work, we attempt to bring both of these approaches—compositionality and state abstraction—together by integrating AI planning (Ghallab et al., 2004) and Reinforcement Learning (Sutton and Barto, 1998). We describe an integrated architecture we call “RePReL,” which combines relational planning (ReP) and reinforcement learning (ReL) in a way that exploits their complementary strengths and not only speeds up the convergence compared to a traditional RL solution but also enables the effective transfer of the skills to multiple tasks and generalization across different numbers of objects (Kokel et al., 2021b,a).

#### 4.1 Introduction

Automated planning and RL have been two major thrusts of AI aimed at sequential decision-making (see §2.4.1–2.4.2). While classical planning focuses on composing sequences of actions offline before execution, RL interleaves planning and execution. Planning is typically associated with domains where dynamics are known and describable, while RL is associated with reactive domains with unknown dynamics. Most prior work in combining planning and RL falls under the general paradigm of “model-based reinforcement learning” (MBRL). Here explicit dynamic models of actions are learned via exploration and used either offline to

compute approximately optimal policies (Brafman and Tennenholtz, 2002; Guestrin et al., 2002) or online in look-ahead search (Silver et al., 2018). Critically, both planning and reinforcement learning components employ the same state-space, while the main motivation for the combination comes from the benefits of efficiency and cost-savings due to offline computation or look-ahead search.

However, in many real-world domains, e.g., driving, the state space of offline planning is rather different from the state space of online execution. Planning typically occurs at the level of deciding the route, while online execution needs to take into account dynamic conditions such as locations of other cars and traffic lights. Indeed, the agent typically does not have access to the dynamic part of the state at the planning time, e.g., future locations of other cars, nor does it have the computational resources to plan an optimal policy in advance that works for all possible traffic events.

The key principles that enable agents to deal with these informational and computational challenges are *abstraction and composition*. In the driving example, while the lower-level state space consists of a more precise location and velocity of the car, the high-level state space consists of abstract, coarse locations such as “O’hare airport”. While the lower-level action space consists of precise actions such as turning the steering wheel by some amount and applying brakes, the high-level actions composite actions such as taking “Exit 205,”. Excepting occasional unforeseen failures, the two levels operate independently of each other and depend on different kinds of information available at different times. This allows the agent to tractably plan at a high level without needing to know the exact state at the time of the execution, and behave appropriately during plan execution by only paying attention to a small dynamic part of the state.

In this chapter, we aim to combine planning and RL with the motivation of exploiting the compositionality in the domain and inducing effective task-specific state abstractions for efficient learning. **An important assumption that we make is the availability of a**

**relational planner designed by a human expert.** If the planner is a hierarchical one, the human expert is used to construct the task-subtask hierarchy, otherwise, the human expert’s guidance is used to define the sequence of high-level tasks for solving the problem. In addition, as we explain later, knowledge from humans is employed to provide safe and effective abstractions.

We make the following key contribution in this chapter: (1) we propose the RePReL architecture that enables multi-level abstractions; (2) we adapt the first-order conditional influence (FOCI) statements (Natarajan et al., 2008, 2005) to determine safe and effective abstractions; (3) we demonstrate the effective transfer of learned skills from one task to another and effective generalization to a different number of objects. Our results in 4 compelling domains show that RePReL significantly outperforms the state-of-the-art Planner+RL combination while achieving better generalization and transfer.

The code and the data used in this chapter are available for public use at the following URL: <https://github.com/starling-lab/RePReL>. The rest of the chapter is organized as follows. Section 4.2 discusses the related work. Then, Section 4.3 describes the RePReL architecture in detail. Section 4.4 describes experiments in 4 different domains that demonstrate generalization and transfer capabilities. Finally, Section 4.5 concludes by summarizing the key insights.

## **4.2 Related work**

### **4.2.1 Planner and RL combination**

The RePReL framework consists of a symbolic planner at the higher level and various RL agents at the ground level. Several prior works have explored the idea of combining a planner and RL agents to solve complex problems which have some notion of temporally extended actions or task hierarchies (Grounds and Kudenko, 2005; Yang et al., 2018; Lyu et al.,

2019; Jiang et al., 2019; Eppe et al., 2019). Among these, RePReL is closely related to the Taskable RL framework of Illanes et al. (2020). Similar to Taskable RL, RePReL employs a planner to generate useful instructions (task definitions) for the RL agent. RePReL extends the Taskable RL framework in three key ways: (1) it generalizes the Taskable RL to solving RMDP, (2) it provides an approach to define task-specific state abstraction in this framework, and (3) it can handle both discrete and continuous domains. Thus, Taskable RL is a natural baseline in our evaluations.

### 4.2.2 Abstraction

Safe and efficient state abstraction techniques have been studied extensively in RL (Li et al., 2006). They have been particularly useful for multi-task and transfer learning problems (Walsh et al., 2006; Sorg and Singh, 2009; Abel et al., 2018). We are inspired by the task-specific, model-agnostic state abstractions of MAXQ (Dietterich, 2000b) and the bisimulation conditions (Ravindran and Barto, 2003; Givan et al., 2003) to define abstractions in relational settings using first-order probabilistic models (Raedt et al., 2016). Andre and Russell (2002b) introduces ALisp language to assert (ir)relevant variable. Our work diverges from their work as we consider the relational setting. Another work by Finzi and Lukasiewicz (2006) leverages situation calculus for explicating the abstraction in a relational setting. We, however, approach this problem using a relational probabilistic model.

## 4.3 Relational planning and Learning

### 4.3.1 Motivational example

In many real-world domains, the exact dynamics of the domain are difficult to provide. However, partial, high-level domain dynamics are readily available from domain experts. Consider a relational taxi domain shown in Figure 4.1①. This domain has one or more

passengers and a taxi. Six actions available in this domain are: **east**, **west**, **north**, **south**, **pick**, **drop**. The goal is to transport passenger(s) from their current location to their destination location. Only 1 passenger can hire the taxi at a time. While it might be difficult to provide complete dynamics of the domain, high-level domain dynamics like “*Taxi can only drop a passenger after pickup*” and “*Only one person can hire the taxi at a time*” are easy to provide. In the RePReL framework, we consider such a class of problems where the high-level domain description is readily available.

We use a high-level planner to first decompose the goal into appropriate subgoals. In the example, the goal of transporting  $p1$  and  $p2$  is decomposed into 4 subgoals: pickup  $p1$ , drop  $p1$ , pickup  $p2$ , and drop  $p2$  by the high-level planner (Figure 4.1②). Different RL agents at a lower level then achieve these subgoals by navigating in the grid and picking up or dropping off the passenger (Figure 4.1③). To decompose the goal into subgoals, the high-level planner does not need the complete grid map. Similarly, RL agents do not need complete information about the state to achieve the goal. For e.g., the RL policy that is performing pickup  $p1$  subgoal needs to know the location of  $p1$  and whether the taxi is free, but passenger  $p2$  and destination of  $p1$  are irrelevant. Similarly, the RL policy performing drop  $p2$  needs to ensure the taxi is hired by  $p2$ , but the pickup location of  $p2$  is irrelevant.

It has been argued that for human-level general intelligence, the two abilities (1) to detect compositional structure in the domain (Lake et al., 2015) and (2) to form task-specific abstractions (Konidaris, 2019) are necessary. In RePReL architecture, we aim to provide these two abilities for efficient task transfer and generalization.

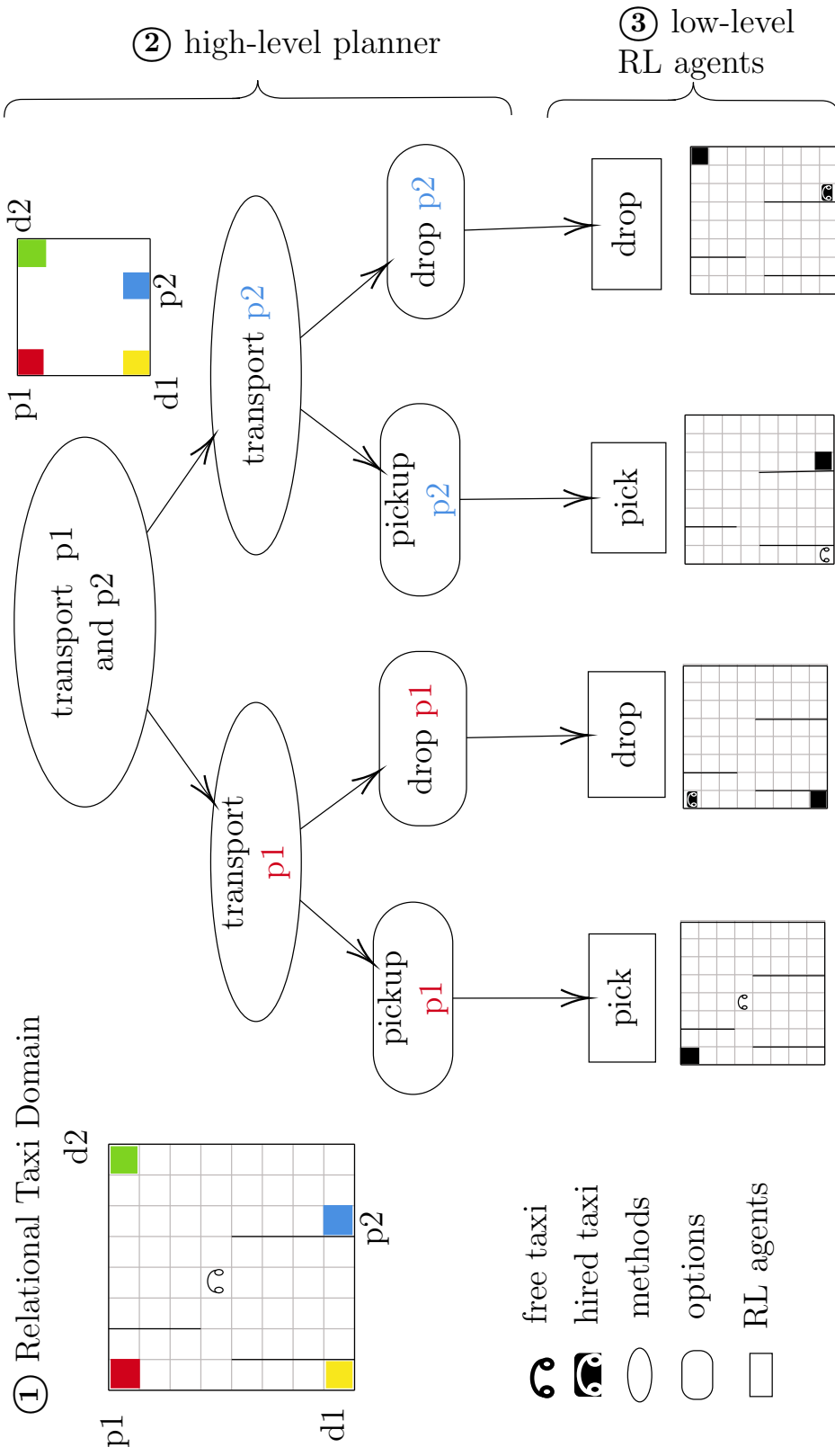


Figure 4.1: Motivational example of RePReL framework in Taxi domain. ① Relational taxi domain. The initial state has two passengers  $p1$  and  $p2$ . The goal is to transport both of them to their respective destination,  $d1$  and  $d2$  respectively. ② The high-level planner decomposes the goal into subgoals by using high-level state representation. ③ Low-level RL agents achieve these subgoals by using abstract state representations.

### 4.3.2 Problem setup

To address a multi-task setting, we first restrict the RMDP definition (see Def. 4) to a *goal-directed RMDP* (GRMDP),  $\mathcal{M} = \langle S, A, T, R, \gamma, G \rangle$ , by introducing a set of goals  $G$  that the agent may be asked to achieve. Different tasks are formulated in a GRMDP by choosing different goals from the set  $G$ . The reward function in this multi-task setting is incognizant of the goal. It provides a domain-specific reward, like a negative reward for invalid actions or step costs. The solution to a GRMDP is a policy  $\pi : S \times G \times A \rightarrow [0, 1]$ , such that when following  $\pi$  from any state  $s$  for goal  $g$  the probability of eventually reaching the goal is 1.

We assume that partial high-level domain knowledge of the problem is specified in the form of a high-level planner. Unlike the classical planning domains  $\mathbf{D} = \langle \mathcal{L}, \mathbf{O} \rangle$  where a schematic operator is a single action, in the high-level planning domain  $\mathcal{D} = \langle \mathcal{L}, \mathcal{O} \rangle$  schematic operators are temporally extended. This is to say that, in classical planning, applying a grounded operator to state results in a single step or transition from one state to another. In high-level planning, applying a grounded operator to a state usually requires multiple steps or multiple subsequent state transitions before it terminates. The termination condition  $\beta$  is also defined for each schematic operator. Given the resemblance to the options framework (Sutton et al., 1998), we refer to these temporally extended operators as *options*. A high-level plan consists of a sequence of grounded options.

We address a class of GRMDPs that combines a high-level symbolic planner with low-level RL policies. Inspired by an earlier work (Illanes et al., 2020), we define this class of GRMDPs as taskable GRMDPs.

**Definition 5.** A GRMDP  $\langle S, A, T, R, \gamma, G \rangle$  is **taskable** if a high-level planning domain  $\mathcal{D}$  can be defined such that all the goals in  $G$  can be composed as some combination of the options in  $\mathcal{D}$ .

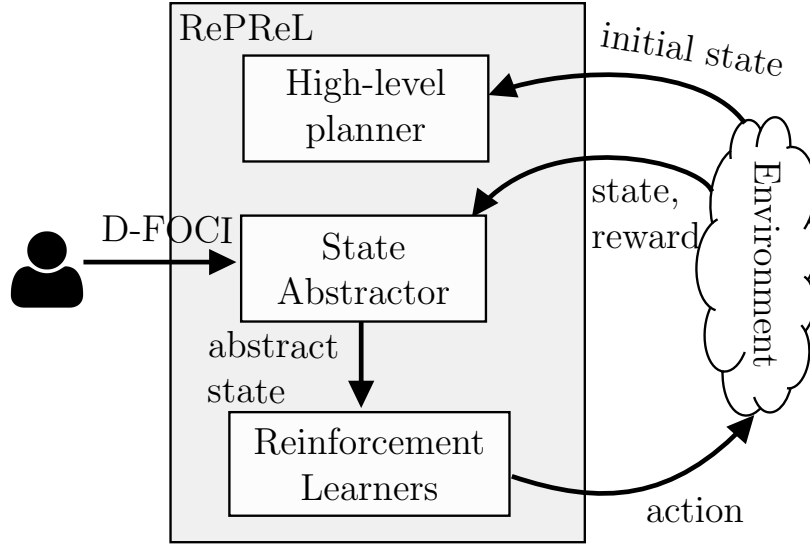


Figure 4.2: RePReL architecture.

The key idea here is that high-level symbolic domain knowledge can be leveraged to identify the compositionality in the domain and individual policies can be learned to solve each composition. *So a solution to a taskable GRMDP is a compositional policy.*

#### 4.3.3 RePReL architecture

Given a taskable GRMDP problem, the RePReL framework obtains the high-level plan from a symbolic planner and then learns RL policies at the low level to achieve each of the options in the high-level plan. The architecture of the RePReL framework is shown in Figure 4.2. It consists of three stacked modules: *symbolic planner*, *state abtractor*, and *reinforcement learners*.

The *symbolic planner* uses the high-level planning domain  $\mathcal{D}$  to decompose the goal into a sequence (or set, if unordered) of grounded options. It must be mentioned that our RePReL framework is **independent of the planner**. If the planner is a hierarchical one, we assume that high-level tasks (methods) are decomposed recursively until low-level options are constructed. If the planner is a non-hierarchical one, the high-level decomposition yields

a set of options. The *state abstractor* generates a task-specific abstract state representation. Finally, multiple *reinforcement learners* at the lowest level learn separate RL policies for each option in the abstract state space. For each option  $o \in \mathcal{O}$  in the high-level planning domain, we define a subgoal RMDP as follows.

**Definition 6.** The **subgoal RMDP**  $M_o$  for an option  $o$  is a tuple  $\langle S, A, P_o, R_o, \gamma \rangle$  consisting of abstract states  $S$ , actions  $A$ , transition function  $P_o$ , reward function  $R_o$ , and discount factor  $\gamma$ . The action space remains the same as for the original GRMDP. The reward function  $R_o$  and transition probability distribution function  $P_o$  are defined as follows:

$$R_o(s, a, s') = \begin{cases} t_R + R(s, a, s') & \text{if } s' \in \beta(o) \text{ \& } s \notin \beta(o) \\ 0 & \text{if } s' \in \beta(o) \text{ \& } s \in \beta(o) \\ R(s, a, s') & \text{otherwise} \end{cases}$$

$$P_o(s, a, s') = \begin{cases} 0 & \text{if } s \in \beta(o) \text{ \& } s' \notin \beta(o) \\ 1 & \text{if } s \in \beta(o) \text{ \& } s' \in \beta(o) \\ P(s, a, s') & \text{otherwise} \end{cases}$$

with  $R$  and  $P$  from the original GRMDP, and a fixed terminal reward  $t_R$ .

This subgoal RMDP is solved by the low-level RL agents in abstract state space. We next describe how the state space is abstracted safely by the state abstractor.

The state abstractor determines the set of state variables that are necessary and sufficient given the current task and provides a task-specific abstract representation of the state for the low-level RL agents. We adopt the bisimulation framework of Givan et al. (2003) and Ravindran and Barto (2003) to define model agnostic abstraction.

**Definition 7.** A *model-agnostic abstraction*  $\phi(s)$  is such that for any action  $a$  and an abstract state  $\bar{s}$ ,  $\phi(s_1) = \phi(s_2)$  if and only if

$$\begin{aligned} \sum_{\{s'_1 | \phi(s'_1) = \bar{s}\}} R_o(s_1, a, s'_1) &= \sum_{\{s'_2 | \phi(s'_2) = \bar{s}\}} R_o(s_2, a, s'_2) \\ \sum_{\{s'_1 | \phi(s'_1) = \bar{s}\}} P_o(s_1, a, s'_1) &= \sum_{\{s'_2 | \phi(s'_2) = \bar{s}\}} P_o(s_2, a, s'_2) \end{aligned}$$

The first condition states that the two states  $s_1$  and  $s_2$  have the same immediate reward distribution with respect to the abstraction, and the second condition states that they have the same transition dynamics. Essentially, the value function of the original MDP is maintained in the abstract MDP. So, the model agnostic abstraction is *safe*. That is, an optimal policy learned with model-agnostic abstraction is also optimal in the original MDP (Li et al., 2006).

To define such a safe model agnostic abstraction function  $\phi$ , we need to identify state literals that neither influence the reward function nor the transition function—they are irrelevant. In this framework, we capitalize on explicit domain knowledge to identify the relevant state literals from irrelevant ones. This domain knowledge is captured using a novel formal language called D-FOCI.

### Why do we need a novel language?

Dietterich (2000a) defines *irrelevant state variables* for an MDP as variables that never influence the reward function or the relevant state variables. Formally,

**Definition 8.** State variables  $Y$  are *irrelevant* in an MDP, if state variables can be partitioned into two disjoint subsets  $X$  and  $Y$  such that

1.  $P(x', y' | x, y, a) = P(x' | x, a)P(y' | x, y, a)$
2.  $R(s, a, s') = R(\langle x, y \rangle, a, \langle x', y' \rangle) = R(x, a, x')$

Given a Dynamic Bayesian Network (DBN) representation (Murphy, 2002) of the transition function of an MDP, the set of relevant variables can be identified by starting at the reward variable and collecting all the variables that influence the collected variables. This set of relevant variables forms a “model-agnostic state abstraction”. Such abstractions ensure that the reward distribution and the transition dynamics in the abstract MDP and the original MDP are the same. Hence, model-agnostic abstraction is safe.

The *transition function of a GRMDP is first-order/relational* and hence a simple DBN does not suffice. Previous works have used a 2-timeslice Probabilistic Relational Model (PRM) (Guestrin et al., 2003) to capture the transition function in relational MDP. However, PRMs are not suitable for capturing GRMDPs. Hence, the RePReL framework used an extension of the First-Order Conditional Influence (FOCI) language (Natarajan et al., 2008) called Dynamic FOCI (D-FOCI).

#### 4.3.4 D-FOCIs

*First-order conditional influence statements* (FOCI), introduced by Natarajan et al. (2005) consist of statements of the form,

$$\text{IF } \langle \text{condition} \rangle \quad \text{THEN} \quad \langle \text{influent} \rangle \quad \text{QINF} \quad \langle \text{resultant} \rangle, \quad (4.1)$$

where **condition** and **influent** are a finite set of literals and **resultant** is a single literal. A FOCI statement encodes the qualitative influence (QINF) of **influent** on the **resultant**. Refer to Section 2.5.1 for further details. *Dynamic-FOCI* (D-FOCI) *statements* extend FOCI statements by adding representation to capture influence over time. The syntax of a D-FOCI statement is as follows,

$$[\langle \text{option} \rangle] : \langle \text{influent} \rangle \xrightarrow{[+1]} \langle \text{resultant} \rangle, \quad (4.2)$$

where the **option** is a temporally extended operator from the high-level planning domain, and **influent** and **resultant** are set of literals. D-FOCI statement states that when executing the given **option**, the **resultant** literal is influenced by literals in **influent**. Following the standard dynamic Bayesian network (DBN) representation of an MDP, we allow action variables in **influent** and reward variables in **resultant**. D-FOCIs denote direct influences between literals in the same time step by an arrow symbol ( $\longrightarrow$ ) and direct influences of the literals in the current time step on the literals in the next time step by a ‘+1’ symbol above the arrow. Non-mandatory components of the D-FOCI statement are denoted within square brackets  $[ ]$ . Omitting the **option** encodes the influences between literals is perpetual (for example, Equation 4.3a encodes that the **action** and **taxi\_at** always influence **taxi\_at**, regardless of the option).

#### 4.3.5 Example of D-FOCIs

In the relational taxi domain (Figure 4.1①), the location of the taxi is influenced by its previous location and the action performed, which is captured in the D-FOCI statement in Equation 4.3a<sup>1</sup>. Further, when executing the task of picking up a passenger, if we assume that the taxi is going to be empty, then we can safely say that only the passenger’s location, the passenger in the taxi, and the taxi’s location influence the completion of the task. This influence is captured in Equations 4.3b and 4.3c. Similarly, the influence information while dropping passenger  $P$  is captured in Equation 4.3d–4.3f.

---

<sup>1</sup>Variables are uppercase. Constants and predicates are lowercase.  $X, Y, D, K$  are variables for location and  $P$  a variable for passenger.

$$\{\text{action}, \text{taxi\_at}(X)\} \xrightarrow{+1} \text{taxi\_at}(X) \quad (4.3a)$$

$$\text{pick}(P) : \{\text{action}, \text{taxi\_at}(X), \text{at}(P, Y), \text{in\_taxi}(P)\} \xrightarrow{+1} \text{in\_taxi}(P) \quad (4.3b)$$

$$\text{pick}(P) : \{\text{in\_taxi}(P)\} \longrightarrow \text{Reward} \quad (4.3c)$$

$$\text{drop}(P) : \{\text{at\_dest}(P)\} \longrightarrow \text{Reward} \quad (4.3d)$$

$$\text{drop}(P) : \{\text{at}(P, X), \text{dest}(P, D), \text{at\_dest}(P)\} \longrightarrow \text{at\_dest}(P) \quad (4.3e)$$

$$\text{drop}(P) : \{\text{action}, \text{taxi\_at}(X), \text{at}(P, Y), \text{in\_taxi}(P)\} \xrightarrow{+1} \text{at}(P, K) \quad (4.3f)$$

#### 4.3.6 Abstraction using D-FOCI

The state abstractor in the RePReL architecture (Figure 4.2) uses D-FOCIs to generate model-agnostic state abstractions. A task-specific model-agnostic abstract representation of the state can be derived from D-FOCI by recursively unrolling and collecting the state literals that influence the relevant state literals, starting from the reward variables. Table 4.1 illustrates this recursive unrolling process.

Given a state  $s$  and a grounded option, the unrolling process begins by grounding the relevant D-FOCI statements<sup>2</sup> that have the *Reward* variable on the right-hand side (RHS). To ground a D-FOCI statement, a substitution  $\theta$  is identified that unifies the literals on the left-hand side (LHS) with state  $s$ . Then the literals in the LHS are collected in a *relevant literals set*  $\hat{s}$ . Then the relevant D-FOCI statements that have RHS in  $\hat{s}$  are grounded and substitution  $\theta$  is refined and literals on LHS are added to  $\hat{s}$ . Note that the substitution is a superset of the previous substitution, that is the existing variable assignment remains the same. This process is repeated recursively. The recursion ends at a fixed depth or when no new literals can be added to  $\hat{s}$ .

---

<sup>2</sup>with matching or null option

Table 4.1: Illustrative example of recursive unrolling of the D-FOCI statements in taxi-domain.

<p><b>Given:</b></p> <ul style="list-style-type: none"> <li>a. D-FOCI statements from Equation 4.3</li> <li>b. state <math>s = \{ \text{at}(p1, r), \text{taxi\_at}(l3), \text{dest}(p1, d1), \neg \text{at\_dest}(p1), \neg \text{in\_taxi}(p1), \text{at}(p2, b), \neg \text{at\_dest}(p2), \neg \text{in\_taxi}(p2) \}</math></li> <li>c. grounded option <math>\theta</math>: <math>\text{pick}(P) \{P/p1\}</math></li> </ul> <p><b>Output:</b> A set of relevant state literals: <math>\hat{s}</math></p>
<p><b>Depth 1 unrolling:</b></p> <ul style="list-style-type: none"> <li>1. Find a substitution that grounds relevant D-FOCI statements that have reward on RHS <ul style="list-style-type: none"> <li><math>\text{pick}(p1): \text{in\_taxi}(p1) \longrightarrow \text{Reward}</math></li> <li><math>\theta = \{P/p1\}</math></li> </ul> </li> <li>2. Collect LHS in relevant literals set <math>\hat{s}</math> <ul style="list-style-type: none"> <li><math>\hat{s} \leftarrow \{ \text{in\_taxi}(p1) \}</math></li> </ul> </li> </ul>
<p><b>Depth 2 unrolling:</b></p> <ul style="list-style-type: none"> <li>1. Find a substitution that grounds relevant D-FOCI statements that have a relevant literal on RHS <ul style="list-style-type: none"> <li><math>\text{pick}(P): \{ \text{action}, \text{taxi\_at}(l3), \text{at}(p1, r), \text{in\_taxi}(p1) \} \longrightarrow \text{in\_taxi}(p1)</math></li> <li><math>\theta = \{P/p1, X/l3, Y/r\}</math></li> </ul> </li> <li>2. Collect LHS in set <math>\hat{s}</math> <ul style="list-style-type: none"> <li><math>\hat{s} \leftarrow \{ \text{in\_taxi}(p1), \text{action}, \text{taxi\_at}(l3), \text{at}(p1, r) \}</math></li> </ul> </li> </ul>
<p><b>Depth 3 unrolling:</b></p> <ul style="list-style-type: none"> <li>1. Ground applicable D-FOCI statements that have a relevant literal (<math>\hat{s}</math>) on RHS <ul style="list-style-type: none"> <li><math>\{ \text{action}, \text{taxi\_at}(l3) \} \xrightarrow{+1} \text{taxi\_at}(l3)</math></li> <li><math>\text{pick}(p1): \{ \text{action}, \text{taxi\_at}(l3), \text{at}(p1, r), \text{in\_taxi}(p1) \} \longrightarrow \text{in\_taxi}(p1)</math></li> <li><math>\theta = \{P/p1, X/l3, Y/r\}</math></li> </ul> </li> <li>2. Collect LHS in set <math>\hat{s}</math> <ul style="list-style-type: none"> <li><math>\hat{s} \leftarrow \{ \text{in\_taxi}(p1), \text{action}, \text{taxi\_at}(l3), \text{at}(p1, r) \}</math></li> </ul> </li> </ul>

While the planner works in relational representations, reinforcement learning operates at a propositional level. The gap is bridged by computing an appropriate propositional abstraction of the state for each operator with the parameters, e.g.,  $p1$ , bound to generic objects (Skolem constants in logic). To keep the size of the propositional representation bounded, we bound the depth of the inference chain through D-FOCI statements to  $k$ . We limit this unrolling process to at most  $k = 2$  levels and at most 1 time step to keep the size of the propositionalization bounded. The set of all such literals forms the final abstraction. The unrolling depth will impact the time complexity of computing the abstractions.

**Theorem 1.** *If the MDP satisfies the D-FOCI statements with a fixed depth unrolling, then the corresponding model-agnostic abstraction has the same optimal value function as the fully instantiated MDP.*

**Proof (sketch):** Fixed depth grounding of the D-FOCI statements would create a propositional DBN with maximum width equal to the depth. If the MDP satisfies the influence information in this DBN then collecting all the variables influencing the reward and the relevant variables provides model agnostic abstraction (Ravindran and Barto, 2003). The grounding and fixed-depth unrolling of the D-FOCI statement begins with the reward variable and collects all the grounded literals influencing the reward and the relevant grounded literals. So the set of grounded state literals collected by this process is identical to collecting all the relevant state variables in the aforementioned propositional DBN. Hence, the grounding and unrolling provide a model-agnostic abstraction.  $\square$

### RePreL, so far

In this problem setup, it is assumed that a high-level symbolic planner is available that can decompose the goal into a high-level plan, i.e. sequence of grounded option, as described in Section 4.3.2. In RePreL architecture, the initial state of the world is provided as

input to a symbolic planner and a high-level plan is obtained as its output. This high-level plan is executed by different RL agents at the lower level. A separate RL agent is trained for each option. The MDP of each RL agent is defined in Definition 6. For each RL agent, the state abstractor provides a task-specific abstraction, as described in Section 4.3.6. The state abstractor uses the D-FOCI statements to derive model-agnostic abstraction. The next section explains how the RePReL architecture is learned.

---

**Algorithm 2** RePReL Learning Algorithm

---

INPUT: env, goal  $g$ , domain  $\mathcal{D} = \langle \mathcal{L}, \mathcal{O} \rangle$ , terminal reward  $t_R$ , D-FOCI statements  $F$ ,  
OUTPUT: RL policies  $\pi_o, \forall o \in \mathcal{O}$

```

1:  $\pi_o, \forall o \in \mathcal{O}$  ▷ initialize RL policy for each operator
2: for each episode do
3:    $s \leftarrow$  get state from env
4:    $\Pi \leftarrow \text{getPlan}(s, g, \mathcal{D})$ 
5:   for  $o\theta$  in  $\Pi$  do
6:      $\pi \leftarrow \pi_o$  ▷ get resp. RL policy
7:      $\hat{s} \leftarrow \text{GetAbstractState}(s, o\theta, F)$ 
8:     done  $\leftarrow \hat{s} \in \beta(o\theta)$  ▷ check terminal state
9:     while not done do
10:       $a \leftarrow \pi(\hat{s})$  ▷ get action
11:       $s' \leftarrow \text{env.step}(a)$  ▷ take a step in env
12:       $r \leftarrow R(s, a, s')$  ▷ get step reward
13:       $\hat{s}' \leftarrow \text{GetAbstractState}(s, o\theta, F)$ 
14:      done  $\leftarrow \hat{s}' \in \beta(o\theta)$  ▷ check terminal next state
15:      if done then
16:         $r = r + t_R$  ▷ add terminal reward
17:      end if
18:       $\pi.\text{update}(\hat{s}, a, \hat{s}', r)$  ▷ update policy
19:       $s, \hat{s} \leftarrow s', \hat{s}'$ 
20:    end while
21:  end for
22: end for
23: return  $\pi_o, \forall o \in \mathcal{O}$ 

```

---

### 4.3.7 Learning

Given the GRMDP environment  $env$ , planner  $\mathfrak{P}$  and D-FOCI statements  $F$ , we now discuss the RePreL learning procedure from **Algorithm 2**. First, for each operator, an RL policy is initialized in **line 1**. Next for each episode, in **line 4**, we get the high-level plan  $\Pi$  from the planner  $\mathfrak{P}$ . For every option in the plan, we train the RL policy  $\pi_o$  (**lines 6–20**). To train the RL policy, we get an abstract propositional state representation  $\hat{s}$  from state  $s$  as described in previous paragraphs. In **lines 10–19**, we obtain action  $a$  from the current policy, perform that action, observe the next state  $s'$  and reward. If  $s'$  is a terminal state for the ground operator  $o\theta$ , then we add a terminal reward  $t_R$  (**line 16**) before updating the policy/q-value (**line 18**). In our experiments, we employ tabular q-learning over the propositional state space for updating the values and consequently, the policy. We repeat the process for a fixed budget of episodes for our evaluation, but various other stopping criteria can also be used.

The low-level RL agents learn optimal policies for the subgoal RMDP. As the original GRMDP is decomposed into the subgoal RMDP by the high-level planner, the resulting RePreL agent is recursively optimal (Dietterich, 1998) assuming the high-level planner and the GRMDP decompositions are optimal. Note that recursively optimal policies are not necessarily globally optimal. Rather, recursively optimal policies ensure that lowest-level policies are locally optimal for the assigned task without any context of the high-level task (Dietterich, 1998). This makes them ideal for multi-task and transfer-learning setups.

## 4.4 Experiments

We now empirically evaluate the RePreL framework by comparing it with tabular Q-Learning (QL), hierarchical RL (HRL), and taskable RL (TRL). All our experiments aim at assessing the sample efficiency and effectiveness of transfer across tasks and generalization

across objects. We aggregate results over 5 runs with different random seeds. We employ the Pyhop planner<sup>3</sup> in our experiments. We aim to answer the following questions.

**Q1. Sample Efficiency:** Do the abstractions induced in RePReL improve sample efficiency?

**Q2. Transfer:** Do these abstractions allow for effective transfer?

**Q3. Generalization:** Does RePReL efficiently generalize to varying number of objects?

#### 4.4.1 Domains

We evaluate our approach on four multi-task domains with discrete state and action spaces. The D-FOCI statements used in each domain are presented in Table 4.2

##### 1. Craft World

This is a Minecraft-inspired multitask grid-world from Andreas et al. (2017). Figure 4.3a presents an example state. It is an  $11 \times 11$  grid with 8 points of interest, grass, wood, iron, gold, gem, workbench, toolshed, and factory, as highlighted in Figure 4.3a. There are four tasks in this environment: **1.** get wood and iron, **2.** make sticks, **3.** make axe, **4.** mine gem. These tasks can be viewed as *curriculum learning* since they are incremental. To mine the gem, an agent needs an axe, and an axe is built by visiting the toolshed with a stick and iron. Sticks are built by first collecting wood and then visiting the workbench. The environment provides a reward of  $-1$  for every step and 100 for a goal state. We use terminal reward  $t_R = 100$  for each subtask.

---

<sup>3</sup>An HTN planner (Nau et al., 1999) written in python, <https://bitbucket.org/dananau/pyhop>

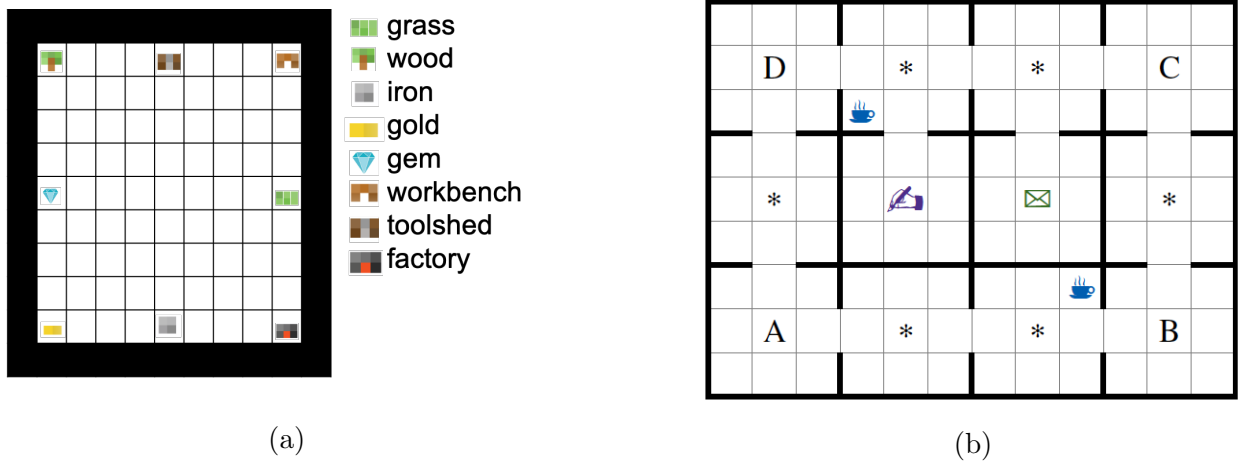


Figure 4.3: (a) Craft World indicating domain with eight locations: grass, wood, iron, gold, gem, workbench, toolshed, and factory. Black cells in the grid represent walls. (b) Office World, reproduced from Illanes et al. (2020).

## 2. Office World

This is a multitask grid-based environment from Illanes et al. (2020). Figure 4.3b reproduces the office world image from Illanes et al. (2020) for convenience. It has inaccessible and accessible locations. Plants are indicated by “\*” in the map, these locations are inaccessible to the agent. Walls in the map are indicated by a bold black line. Any step toward a wall or inaccessible locations will keep the agent in the same location. Accessible locations include two coffee rooms, one office desk, and one mail room. Coffee room locations are indicated by blue coffee mugs, the mail room is indicated by a green envelope, and the office desk is highlighted with a hand. Locations A, B, C, and D are also accessible locations, they are marked in the figure. The four tasks include: **1.** deliver mail to office, **2.** deliver coffee to office, **3.** deliver mail and coffee, and **4.** visit locations A, B, C, D. This environment has a reward of  $-1$  for every step,  $100$  for the goal state,  $-10$  for an attempt of going to inaccessible locations. and terminal reward  $t_R = 100$ .

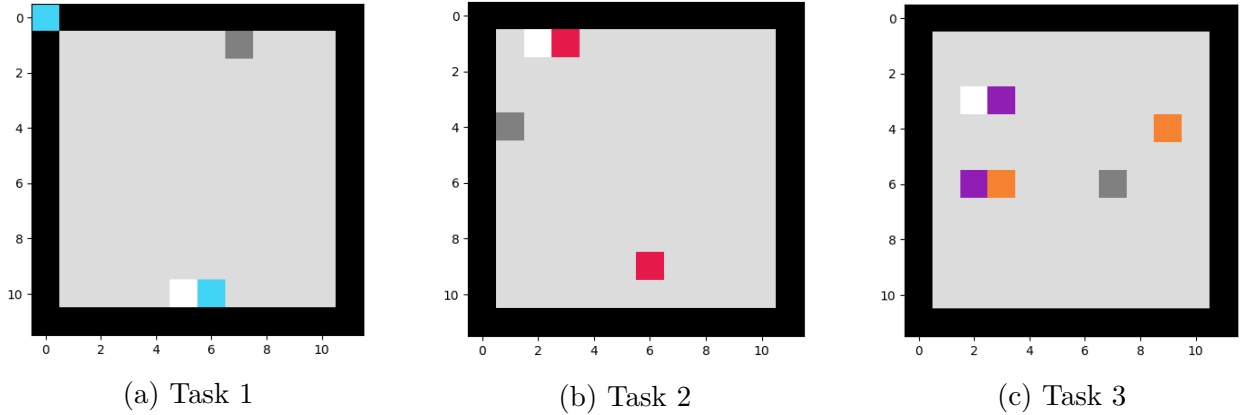


Figure 4.4: Tasks in the Relational Box World.

### 3. Extended Taxi World

We extend the Taxi domain by Dietterich (2000b) with three passengers and the relational representation as shown in the earlier example (Figure 4.1①). The taxi can be hired by one passenger at a time. The task in this environment is to drop all passengers in sequence. Task **1** is to drop one passenger ( $p1$ ), Task **2** is to drop two passengers ( $p1$ ,  $p2$ ), and Task **3** is to drop all three passengers ( $p1$ ,  $p2$ , and  $p3$ ) to their respective destinations in that order. This domain has *higher complexity* than both of the previous domains. In both the Craft and Office World, every object except the agent has a fixed location. Here, along with the agent taxi location, the passenger pick-up and drop locations are also randomly sampled at the beginning of each episode from  $R$ ,  $G$ ,  $B$ , or  $Y$ . Hence, there are 36 possible combinations of passenger pickup and drop locations. Like the office world, every step has a reward of  $-1$ , the goal state has a reward of 100, and the step in an invalid direction has a reward of  $-10$ . We used 100 for the terminal reward.

### 4. Relational Box World

This environment is inspired by Box World from Zambaldi et al. (2019). There are 4 types of objects: lock, key, gem, wall with an associated color. A lock can be opened with a key of

the same color and the player has to open a lock to reach the key inside that box. The player is equipped with sensors on each of its 8 directions ( $NE, E, S, \dots$ ), which detects the relative direction of the objects. Unlike the image representation, in our setting, *the complete grid is not visible to the agent*. The goal, for each task, is to collect the gem. Figure 4.4 represents the three tasks evaluated in the **Relational Box World**. The gem is represented with white color in the grid, the wall with black, and the player with gray. The pair of colored cells represent a box; the cell on right is a lock and the cell on left is either a key or a gem. The owned key is represented on the top right corner of the grid. Locks and keys are sampled from 18 different colors. The locations of the boxes are also sampled at the beginning of each episode. Task 1 has a lock containing the gem, the player is initialized with the key to open the lock. In Task 2 player has to first collect the free key and then open the lock to collect the gem. Task 3 requires the player to collect the key and open two locks in sequence to reach the gem. The environment provides a reward of  $-0.1$  for every step,  $-0.2$  for an invalid step, and 1 for collecting a key or gem. We use terminal reward  $t_R = 1$ .

#### 4.4.2 Sample efficiency

To evaluate the effect of RePReL abstractions, we compare it against the *seq* variant of the Taskable RL. We pick this variant for two reasons: (1.) *seq* variant performed best in all their experiments, (2.) We aim to evaluate the effectiveness of abstractions and thus do not learn the meta-controller introduced by the partially ordered plans. We set a budget on the number of steps for learning in each task and evaluate the performance of RePReL against Taskable RL (TRL), option-based Hierarchical RL (HRL), and q-learning (QL).

Fig. 4.5 presents the performance of RePReL and the baselines in **Craft World** environment with 50K steps budget (consider only the methods without “+” in the graphs. “+T” denotes *with transfer*). All the Figures report aggregated results over 5 runs. Here, TRL and HRL have 8 distinct options, one for each location in the domain. RePReL has two

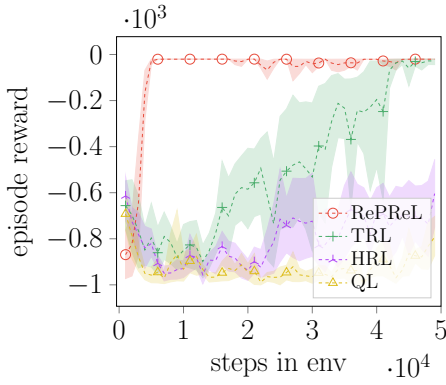
policies; one for each operator shown in Table 4.2. **RePReL significantly outperforms the baselines in all the tasks.**

Figure 4.6 compares the learning curves of RePReL in **Office World** with a  $30K$  budget. While TRL and HRL use 7 different options for traveling to each location in the domain, our RePReL framework uses **only two** options; one each for pickup and deliver operator. We define a common operator for pickup and visit operation as they do not have any pre-conditions and have the same effects. Figure 4.6a shows that RePReL achieves the optimal reward for *Task 1* in less than  $10K$  steps, while the baseline methods TRL and HRL do not achieve optimality even after  $30K$  steps. Similarly in all the other tasks, we see that RePReL consistently outperforms TRL and HRL by converging to the optimal policy in less than  $15K$  steps. Hence, we answer Q1 affirmatively in that RePReL abstractions help in statistically significantly outperforming the state-of-the-art hybrid planner-RL architecture, Taskable RL.

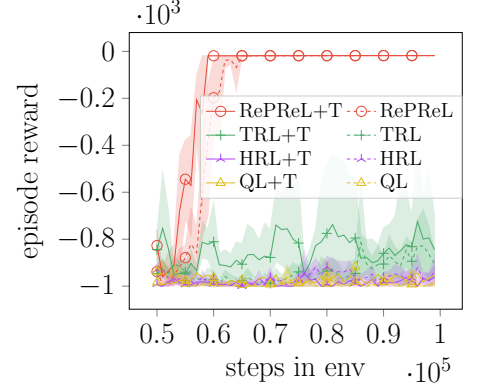
#### 4.4.3 Task transfer

To evaluate the transfer, we modify the RePReL learning algorithm. Specifically, the RL policies are not initialized randomly (line 1 in Algorithm 2), and instead, we transfer the learned policies from *Task 1* to *Task 2*, refine the policy on *Task 2*, transfer it to *Task 3*, and so on, in increasing order. We indicate the transferred RePReL agent performance as RePReL+T in the plots. Similarly, transferred taskable RL and hierarchical RL baselines are indicated by TRL+T and HRL+T respectively. We use the same budget as the previous set of experiments.

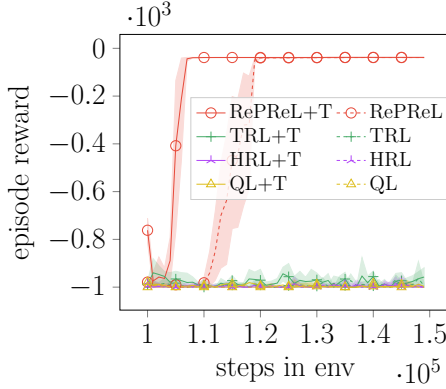
Our experiments (Fig. 4.5 and 4.6) clearly demonstrate that transferring the RePReL policies across different tasks has a distinct advantage in terms of sample efficiency. This advantage is more pronounced in tasks that are closely related to prior tasks. For e.g., in Craft World, tasks are incremental in nature, and consequently, the significant advantage of



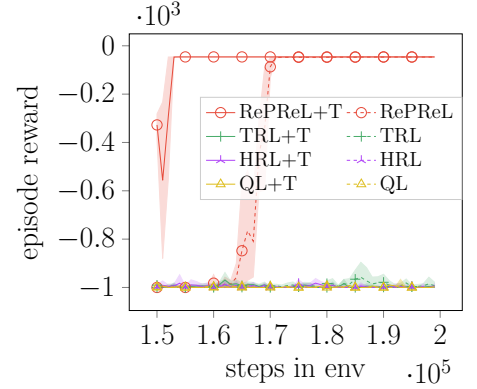
(a) Task 1



(b) Task 2



(c) Task 3

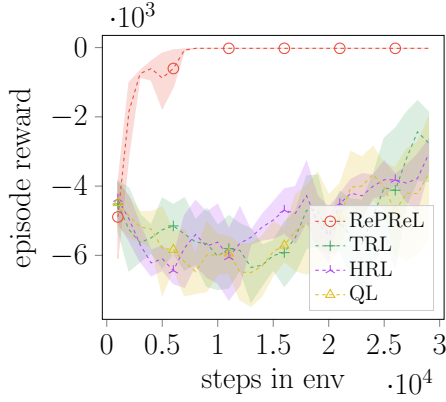


(d) Task 4

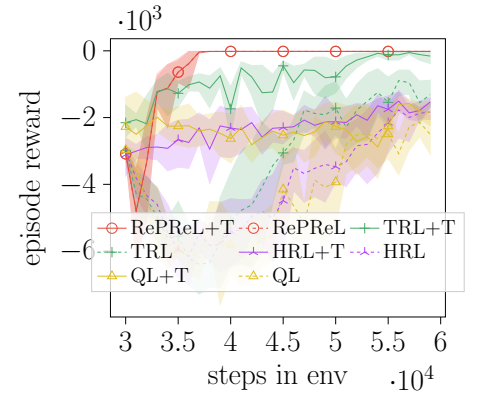
Figure 4.5: Comparing learning curves of RePReL, TRL, HRL, and QL in Craft World environment. Transferred policies are indicated by “+T”. *Task 1* is to get wood and iron, *Task 2* is make stick, *Task 3* is make axe, *Task 4* is mine gem.

RePReL+T over RePReL and other baselines in *Tasks 2, 3, and 4* can be clearly observed. In Office World, the *Task 4* is independent of *Task 1, 2, and 3*, and thus, the gain due to transfer is not significant. Yet, RePReL+T converges faster than TRL+T due to the state abstractions. This allows us to answer Q2 affirmatively.

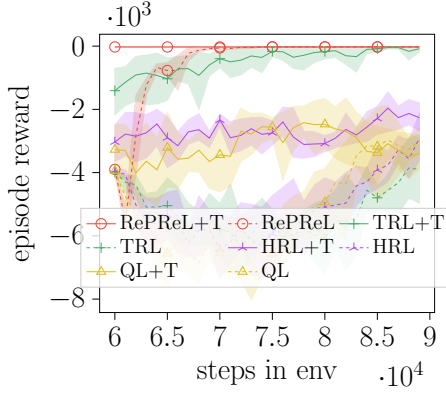
The significant advantage of RePReL over TRL in Craft World when compared to the Office World, is because Craft World has more objects (11 vs 9). **We hypothesize and verify (in the next section) that the advantage of state abstraction is more appar-**



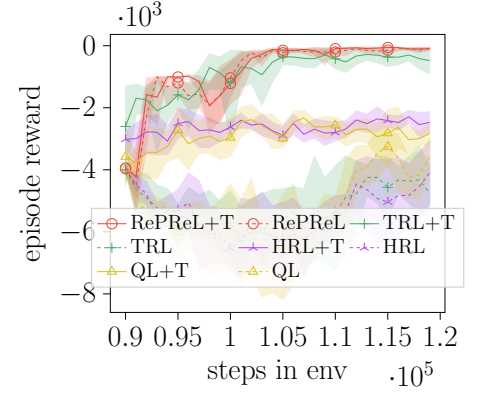
(a) Task 1



(b) Task 2



(c) Task 3



(d) Task 4

Figure 4.6: Comparing learning curves of RePReL, TRL, HRL and QL in Office World environment. Transfer algorithms are indicated by “+T”. *Task 1* is to deliver mail, *Task 2* is to deliver coffee, *Task 3* is to deliver mail and coffee, and *Task 4* is to visit A, B, C, D. Note that the RePReL and RePReL+T curves in Task 2 are overlapping.

ent in the relational domains where the number of objects is higher and effective transfer requires generalization across objects.

#### 4.4.4 Generalization

We present, in Fig. 4.7, the comparison of the RePReL with baselines in the **Extended Taxi World**. Here, both TRL and HRL use 4 options for each location  $R$ ,  $G$ ,  $B$ , and  $Y$ , while RePReL uses **only two options**, one for each operator: pickup and drop a passenger.

The results clearly show that RePReL consistently outperforms both baselines. RePReL with transfer can perform *Task 2* and *Task 3* seamlessly without any additional learning. This demonstrates the generalization capability of the RePReL agent across different passengers.

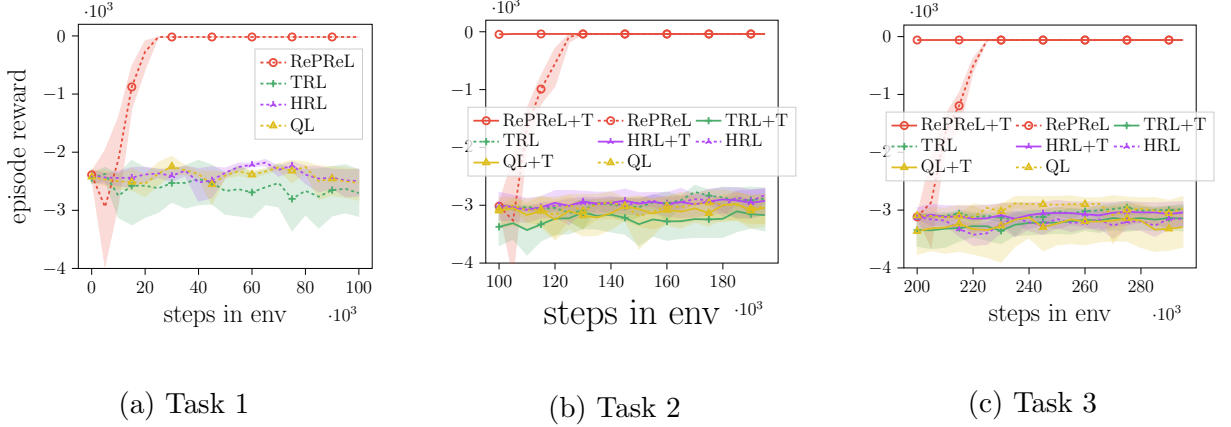


Figure 4.7: Comparing learning curves of RePReL, TRL, HRL, and QL in the Extended Taxi World. *Task 1* is to drop passenger p1, *Task 2* is to drop p1 and p2, *Task 3* is to drop p1, p2, p3.

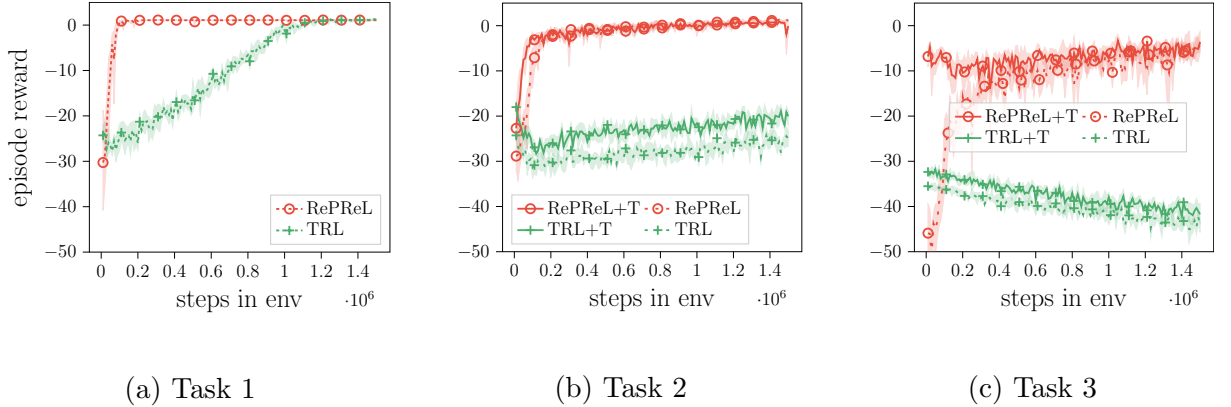


Figure 4.8: Comparing learning curves of RePReL and TRL with and without transfer in Box World environment. The goal in all the tasks is to collect the gem, we increase the number of objects to reach the gem in each task.

Our experiments in the **Relational Box World** domain are presented in Fig. 4.8. Here, we use two subtask policies for both the TRL and RePReL: one for opening a lock and another for collecting a key or a gem. Since the locations of the lock and key are not fixed, we

cannot use different options for each location in taskable RL. Each learning agent is provided a budget of  $1.5M$  for training steps in each task. We see that RePReL is significantly more efficient than TRL, in all three tasks. Here, *Task 2* involves opening one box (i.e. collecting a key and opening a lock) to reach the gem and *Task 3* requires opening two boxes. It can be clearly observed that RePReL+T is able to generalize across a number of objects when going from *Task 2* to *Task 3*. These transfer results in Extended Taxi World and Relational Box World allow us to answer **Q3** affirmatively in that RePReL allows for generalizing across varying numbers of objects and is best suited for relational domains.

We also employed two relational RL baselines: Q-tree (Dzeroski et al., 2001) and Gradient Boosted Q-Learning (Das et al., 2020). However, with the number of time steps that we used for the other planner-based methods, these RRL methods could not converge to an optimal policy. We hypothesize that this is due to the fact that they learn on fixed goal domains while our domains have varying goals (See Appendix B.1). Investigating the extension of RRL to goal-directed methods is an interesting future direction.

## 4.5 Conclusion

We introduced RePReL, a unified framework that enables efficient learning to act in structured domains. The framework consists of three specific components—a high-level planner, a state abstractor, and RL agents. The planner decomposes the tasks into smaller options, the state abstractor computes the appropriate abstractions for the lowest level MDP, and finally, an RL agent learns quickly and effectively given the smaller MDP. Our experiments demonstrate that the RePReL is not only effective and efficient but generalizes to unseen tasks and a larger number of objects.

It is important to note one of the significant differences between the RePReL learning algorithm and the Taskable RL learning algorithm. Taskable RL learns a different policy for each goal location  $R, G, B, Y$  i.e.  $\pi_R(s)$ ,  $\pi_G(s)$ ,  $\pi_B(s)$ ,  $\pi_Y(s)$ , while RePReL learns policy

for pickup and drop operators i.e.  $\pi_{pickup}(s), \pi_{drop}(s)$ . These policies are equivalent (i.e., the action selected by TRL would be the same as RePReL), the difference lies in the way state  $s$  is represented. Taskable RL uses the complete state representation while RePReL generates abstract state representations. Further, RePReL learning algorithm proposed here is that the Taskable RL updates all the subtask policies for every step in the environment, while RePReL only updates the active subtask policy for each step. With these observations, our empirical results should make stronger cases for sample efficiency and generalization abilities for RePReL.

While our empirical evaluation of RePReL is quite successful, the current work has a few limitations. First, it is limited to discrete domains and traditional RL. Second, the current work requires propositional state representation at the low level. Third, RePReL assumes the state spaceflight of the MDPs is structured. Forth, RePReL assumes downward refinement property (Bacchus and Yang, 1991). That is, we assume that a lower-level solution exists for every high-level plan and it can be achieved without backtracking on the high-level plan. Fifth, the current work assumes that the high-level domain description and the D-FOCI statements are specified in advance by the human expert, and most importantly, they are precise and correct. In the following couple of chapters, we address a few of these limitations.

Table 4.2: D-FOCI statments and relevant state predicates for all the domains

Domain	D-FOCI statements	Operators	Set of relevant state predicates
Craft World	$\{\text{agent-at}(\text{L1}), \text{move}(\text{Dir})\} \xrightarrow{+1} \text{agent-at}(\text{L2})$ $\{\text{agent-at}(\text{L1}), \text{move}(\text{Dir})\} \rightarrow R$ $\{\text{with-agent}(\text{Y}), \text{require}(\text{X}, \text{Y})\} \xrightarrow{+1} \text{require}(\text{X}, \text{Y})$ $\text{pickup}(\text{X}); \text{with-agent}(\text{X}) \rightarrow R_o$ $\text{pickup}(\text{X}); \{\text{agent-at}(\text{L1}), \text{at}(\text{X}, \text{L}), \text{with-agent}(\text{X})\} \rightarrow \text{with-agent}(\text{X})$	pickup(X)	$\{\text{agent-at}(\text{L1}), \text{at}(\text{X}, \text{L}), \text{with-agent}(\text{X}), \text{move}(\text{Dir})\}$
	$\text{build}(\text{X}); \{\text{agent-at}(\text{L1}), \text{build-at}(\text{X}, \text{L}), \text{require}(\text{X}, \text{Y}), \text{with-agent}(\text{Y})\} \xrightarrow{+1} \text{with-agent}(\text{X})$ $\text{build}(\text{X}); \text{with-agent}(\text{X}) \rightarrow R_o$	build(X)	$\{\text{agent-at}(\text{L1}), \text{with-agent}(\text{X}), \text{build-at}(\text{X}, \text{L}), \text{require}(\text{X}, \text{Y}), \text{with-agent}(\text{Y}), \text{move}(\text{Dir})\}$
Office World	$\{\text{agent-at}(\text{L1}), \text{move}(\text{Dir})\} \xrightarrow{+1} \text{agent-at}(\text{L2})$ $\{\text{agent-at}(\text{L1}), \text{move}(\text{Dir})\} \rightarrow R$ $\text{pickup}(\text{X}); \{\text{agent-at}(\text{L1}), \text{at}(\text{X}, \text{L}), \text{with-agent}(\text{X})\} \xrightarrow{+1} \text{with-agent}(\text{X})$	pickup(X)	$\{\text{agent-at}(\text{L1}), \text{at}(\text{X}, \text{L}), \text{with-agent}(\text{X}), \text{move}(\text{Dir})\}$
	$\text{pickup}(\text{X}); \text{with-agent}(\text{X}) \rightarrow R_o$ $\text{deliver}(\text{X}); \{\text{agent-at}(\text{L1}), \text{with-agent}(\text{X}), \text{office}(\text{L}), \text{delivered}(\text{X})\} \xrightarrow{+1} \text{delivered}(\text{X})$ $\text{deliver}(\text{X}); \text{delivered}(\text{X}) \rightarrow R_o$	deliver(X)	$\{\text{agent-at}(\text{L1}), \text{with-agent}(\text{X}), \text{office}(\text{L}), \text{move}(\text{Dir}), \text{delivered}(\text{X})\}$
Exte- nded Taxi	$\{\text{taxi-at}(\text{L1}), \text{move}(\text{Dir})\} \xrightarrow{+1} \text{taxi-at}(\text{L2})$ $\{\text{taxi-at}(\text{L1}), \text{move}(\text{Dir})\} \rightarrow R$ $\text{pickup}(\text{P});$ $\{\text{taxi-at}(\text{L1}), \text{at}(\text{P}, \text{L}), \text{in-taxi}(\text{P})\} \xrightarrow{+1} \text{in-taxi}(\text{P})$ $\text{pickup}(\text{P}); \text{in-taxi}(\text{P}) \rightarrow R_o$ $\text{drop}(\text{P}); \{\text{taxi-at}(\text{L1}), \text{in-taxi}(\text{P}), \text{dest}(\text{P}, \text{L}), \text{at-dest}(\text{P})\} \xrightarrow{+1} \text{at-dest}(\text{P})$ $\text{drop}(\text{P}); \text{at-dest}(\text{P}) \rightarrow R_o$	pickup(P)	$\{\text{taxi-at}(\text{L1}), \text{at}(\text{P}, \text{L}), \text{in-taxi}(\text{P}), \text{move}(\text{Dir})\}$
	$\text{drop}(\text{P}); \text{at-dest}(\text{P}) \rightarrow R_o$	drop(P)	$\{\text{taxi-at}(\text{L1}), \text{in-taxi}(\text{P}), \text{dest}(\text{P}, \text{L}), \text{at-dest}(\text{P}), \text{move}(\text{Dir})\}$
Relat- ional Box World	$\{\text{neighbor}(\text{Dir}, \text{C}), \text{agent-at}(\text{L2}), \text{move}(\text{D})\} \xrightarrow{+1} \text{agent-at}(\text{L1})$ $\{\text{neighbor}(\text{Dir}, \text{C}), \text{agent-at}(\text{L1}), \text{move}(\text{D})\} \rightarrow R$ $\text{pickkey}(\text{K}); \text{own}(\text{K}) \rightarrow R_o$ $\text{pickkey}(\text{K}); \{\text{agent-at}(\text{L1}), \text{direction}(\text{K}, \text{Dir2}), \text{own}(\text{K})\} \xrightarrow{+1} \text{own}(\text{K})$	pickkey(K)	$\{\text{neighbor}(\text{Dir}, \text{C}), \text{agent-at}(\text{L1}), \text{direction}(\text{K}, \text{Dir2}), \text{own}(\text{K}), \text{move}(\text{D})\}$
	$\text{unlock}(\text{L}); \text{open}(\text{L}) \rightarrow R_o$ $\text{unlock}(\text{L}); \{\text{agent-at}(\text{L1}), \text{direction}(\text{L}, \text{Dir2}), \text{open}(\text{L})\} \xrightarrow{+1} \text{open}(\text{L})$	unlock(L)	$\{\text{neighbor}(\text{Dir}, \text{C}), \text{agent-at}(\text{L1}), \text{direction}(\text{L}, \text{Dir2}), \text{open}(\text{L}), \text{move}(\text{D})\}$

## CHAPTER 5

### REPREL EXTENSION TO DEEP, NEURAL RL

In Chapter 4, we proposed an integrated planner and RL architecture—RePreL—that uses task-specific state abstractions from the D-FOCI statements. However, we only handled discrete domains in that chapter, with tabular Q-learning. In this chapter, we extend the formalism and present a batch-learning approach that is compatible with deep RL and handles both discrete and continuous states and actions (Kokel et al., 2022b,a).<sup>1</sup> Further, while Chapter 4 generated the ground state abstractions by unrolling the D-FOCI to a fixed depth (which limits its use), in this chapter we allow for richer recursion and relational state representations by employing graph neural networks.

The rest of the chapter is organized as follows. Section 5.1 proposes the deep extension for the RePreL framework. Section 5.2 evaluates the proposed deep RePreL algorithm with Double Deep Q-Network (van Hasselt et al., 2016) and Soft-Actor Critic (SAC) (Haarnoja et al., 2018) as base learners. Finally, Section 5.3 concludes by summarizing the key insights. The code used in this chapter is available for public use at the following URL: <https://github.com/starling-lab/DeepRePreL>.

#### 5.1 Deep RePreL

To extend the RePreL framework to deep, neural reinforcement learning methods, we use neural reinforcement learners at the low level. Essentially, replacing the Reinforcement Learners in the RePreL architecture (Figure 4.2) with Deep Reinforcement Learners. The updated RePreL architecture is shown in Figure 5.1. The updated architecture cannot be

---

<sup>1</sup>The work appearing in this chapter was first published in Neural Computing and Applications, 2022 by Springer Nature. The Version of Record is available online at: <https://doi.org/10.1007/s00521-022-08119-y>

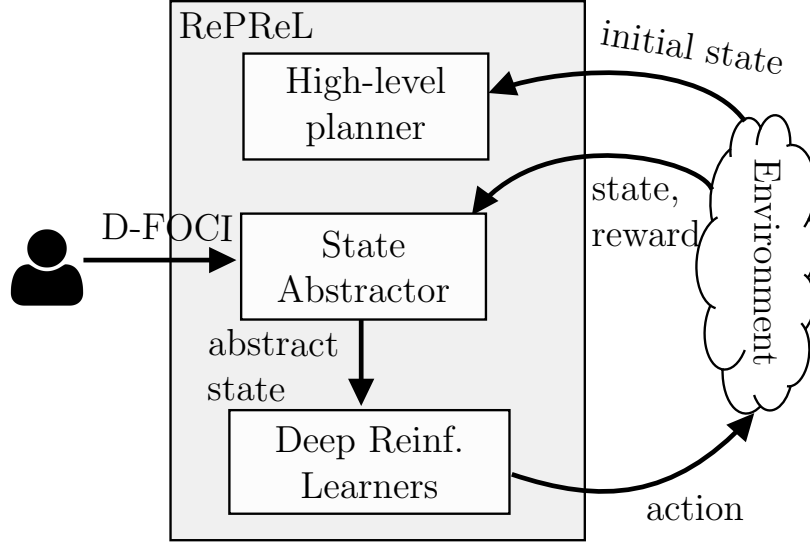


Figure 5.1: RePreL architecture.

learned in an online fashion and, hence, requires a batch-learning algorithm. The modified RePreL Batch Learning algorithm is presented next. We use the same notations as Chapter 4.

### 5.1.1 Batch learning

Given a taskable GRMDP environment  $\text{env}$  with the high-level planning domain  $\mathcal{D}$  and the D-FOCI statements  $F$ , we now discuss the batch learning procedure of the RePreL presented in **Algorithm 3**. First, for each option, an RL policy  $\pi_o$  and a replay buffer  $\mathcal{B}_o$  are initialized in **line 1**. Next, for the current instance of the GRMDP problem, a high-level plan  $\Pi$  is obtained from the planner in **line 5**. For every grounded option in the plan, training samples are collected in the respective buffer  $\mathcal{B}_o$  (**lines 6–22**). An abstract state representation  $\hat{s}$  is obtained in **line 8** and if it is not a terminal state then samples are collected in the buffer till a terminal state is reached (**lines 10–21**). To collect samples, an action  $a$  is obtained from the current policy, that action is performed, and the next state  $s'$  and the reward  $r$  are observed. If  $s'$  is a terminal state for the ground option  $o\theta$ , then a terminal reward  $t_R$  is

added (**line 17**) before pushing it to the buffer (**line 19**). Once enough samples are collected in the buffer, the option policy  $\pi_o$  is updated for each option by sampling a batch from buffer  $\mathcal{B}_o$  (**lines 25–28**). The process is repeated for a fixed budget of episodes during evaluation but various other stopping criteria can also be used.

---

**Algorithm 3** Deep RePreL Batch Learning Algorithm

---

INPUT: **env**, goal  $g$ , domain  $\mathcal{D} = \langle \mathcal{L}, \mathcal{O} \rangle$ , terminal reward  $t_R$ , D-FOCI statements  $F$ , num of iterations  $i$ , num of episodes in each iteration  $k$ , batch size  $b$   
 OUTPUT: RL policies  $\pi_o, \forall o \in \mathcal{O}$

```

1:  $\pi_o, \mathcal{B}_o, \forall o \in \mathcal{O}$  ▷ initialize a policy and a buffer for each option
2: for iteration  $\in i$  do
3:   for episode  $\in k$  do
4:      $s \leftarrow$  get state from env
5:      $\Pi \leftarrow \text{getPlan}(s, g, \mathcal{D})$ 
6:     for  $o\theta$  in  $\Pi$  do
7:        $\pi \leftarrow \pi_o$  ▷ get resp. policy
8:        $\hat{s} \leftarrow \text{GetAbstractState}(s, o\theta, F)$ 
9:        $\text{done} \leftarrow \hat{s} \in \beta(o\theta)$  ▷ check terminal state
10:      while not done do
11:         $a \leftarrow \pi(\hat{s})$  ▷ get action
12:         $s' \leftarrow \text{env.step}(a)$  ▷ take a step in env
13:         $r \leftarrow R(s, a, s')$  ▷ get step reward
14:         $\hat{s}' \leftarrow \text{GetAbstractState}(s, o\theta, F)$ 
15:         $\text{done} \leftarrow \hat{s}' \in \beta(o\theta)$  ▷ check terminal next state
16:        if done then
17:           $r = r + t_R$  ▷ add terminal reward
18:        end if
19:         $\mathcal{B}_o \leftarrow \mathcal{B}_o \cup \{\hat{s}, a, r, \hat{s}', o\theta\}$  ▷ push to the buffer
20:         $s, \hat{s} \leftarrow s', \hat{s}'$ 
21:      end while
22:    end for
23:  end for
24:  for  $o \in \mathcal{O}$  do ▷ Update all the policies
25:     $\pi_o \leftarrow \text{UpdatePolicy}(\pi_o, \text{SampleBatch}(\mathcal{B}_o, b))$ 
26:  end for
27: end for
28: return  $\pi_o, \forall o \in \mathcal{O}$ 

```

---

With batch learning, the RePReL framework can support any off-policy RL algorithm. Additionally, the batch learning algorithm can be adapted to any of the sampling and replay strategies to speed up the training. In our evaluations, we employ a hindsight experience replay buffer (Andrychowicz et al., 2017) with Soft Actor-Critic (Haarnoja et al., 2018) learner.

### 5.1.2 Recursive abstraction with DRRL

The batch learning algorithm is also compatible with the DRRL approaches (Zambaldi et al., 2019; Li et al., 2020; Jiang et al., 2021). DRRL approaches use graph-based neural networks and support relational state representation that allow varying number of objects across states (see §2.4.1). By using DRRL as low-level learners, RePReL can use a relational representation of the state at the low-level. This eliminates the need for fixed object representations in the abstract state. Hence, RePReL can now allow for varying the unrolling depth.

The recursive abstraction process for the relational representation remains the same as described in Section 4.3.6. The recursion ends only when no new literals can be added to  $\hat{s}$ .

**Theorem 2.** *If the MDP satisfies the influence information of the D-FOCI statements then the above procedure that recursively collects the set of state literals that influence the relevant state literals and reward variables is a model agnostic abstraction.*

**Proof (sketch):** Complete grounding of the D-FOCI statements would create a propositional DBN with all the grounded literals as variables. If the MDP satisfies the influence information in this DBN then collecting all the variables influencing the reward and the relevant variables provides model agnostic abstraction (Ravindran and Barto, 2003). The recursive grounding and unrolling of the D-FOCI statement begins with the reward variable and collects all the grounded literals influencing the reward and the relevant grounded literals. So the set of grounded state literals collected by this process is identical to collecting all

the relevant state variables in the propositional DBN. Hence, the recursive grounding and unrolling provide a model-agnostic abstraction.  $\square$

## 5.2 Experiments

We now empirically evaluate our approach on four multi-task domains with discrete as well as continuous state and action spaces. We compared the RePReL framework with traditional tabular RL, tabular hierarchical RL, and tabular taskable RL approaches in Chapter 4. Here, we compare the Deep RePReL framework with double deep Q-Network (DQN) (van Hasselt et al., 2016), hierarchical DQN (HDQN) (Kulkarni et al., 2016), deep taskable RL, and deep relational RL. We consider *two different representations*:

1. **Tensor**, we use tensor representation to evaluate RePReL with double DQN as the base learner.
2. **Graph**, we use graph representation to evaluate RePReL with deep relational RL-based learners.

The high-level planner remains the *same* for all these representations. This demonstrates the **generality** of our proposed framework.

All the experiments aim at assessing the sample efficiency and effectiveness of transfer across tasks and generalization across objects. The results are aggregated over 5 runs with different random seeds. We employ the Pyhop planner<sup>2</sup> in our experiments. We aim to demonstrate the following key aspects of the proposed framework:

- A1.** RePReL framework demonstrates a significant advantage in sample efficiency.
- A2.** RePReL framework facilitates effective transfer across tasks.

---

<sup>2</sup>An HTN planner (Nau et al., 1999) written in python, <https://bitbucket.org/dananau/pyhop>

- A3.** RePReL framework efficiently generalizes across multiple objects.
- A4.** RePReL handles both discrete and continuous state-action spaces, making it a versatile framework for learning in real domains.
- A5.** RePReL batch learning algorithm adapts easily to different off-policy deep RL algorithms.

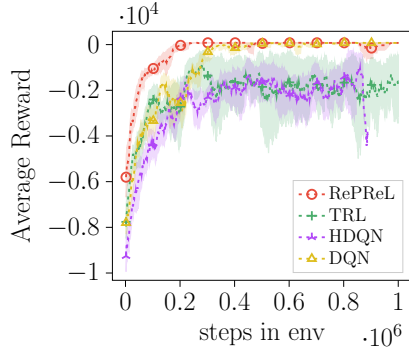
### 5.2.1 Deep RL

In this section we evaluate deep RePReL on two domains Office World and Extended Taxi World from Chapter 4. We used *Task 1* and *2* of Office World to evaluate the transfer efficiency and *Task 1–3* of the Extended Taxi World to evaluate generalization capability. We use double DQN implementation available in the RL-kit package<sup>3</sup> as base learners. We compare our deep RePReL agent against a double *Deep Q-Network (DQN)*, a *Hierarchical DQN (HDQN)*, and a *Taskable-RL (TRL)* agent. Hyperparameters and network architecture were tuned for the DQN agent and used verbatim for other agents. These hyperparameter values are summarized in Table 5.1 and D-FOCI statements are presented in Table 5.3. We set a budget of  $1e6$  on the number of steps that can be taken in the training environment for learning in each task.

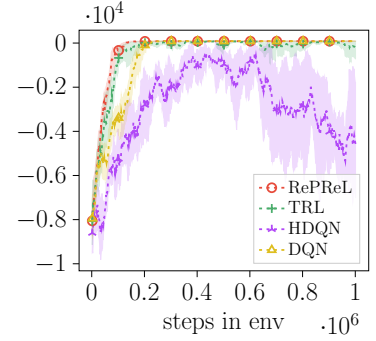
Figures 5.2 and 5.3 compare the learning curves of the four agents on two tasks in Office World and three tasks in Extended Taxi World. While the RePReL, TRL and DQN agents achieve comparable performance in *Task 2* of Office world (Fig. 5.2b) and *Task 1* of Extended Taxi World (Fig. 5.3a), their performance significantly differs in the remaining tasks. Solid lines with “+T” are transferred agents. We see that all the transferred agents start at a higher average reward, but the RePReL agent has the steepest learning curve. Hence, Figures 5.2 and 5.3 demonstrate sample efficiency (A1), effective transfer (A2), and generalization across objects (A3) respectively.

---

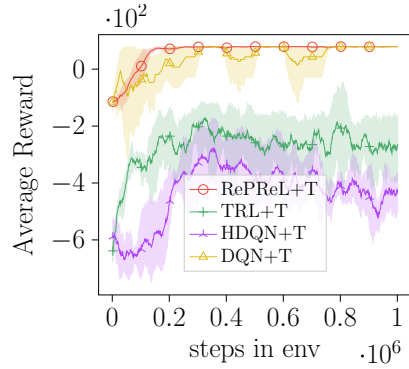
<sup>3</sup><https://github.com/rail-berkeley/rlkit>



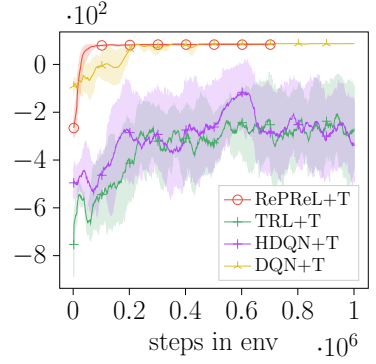
(a) Task 1



(b) Task 2



(c) Transfer to Task 1



(d) Transfer to Task 2

Figure 5.2: Comparing learning curves of deep RL based learners in the Office World environment. RePreL compared against TRL, HDQN and DQN in (a) *Task 1* deliver mail and (b) *Task 2* deliver coffee. Agents are swapped after  $1e6$  steps. RePreL+T compared against TRL+T, HDQN+T and DQN+T in (c) *Task 1* and (d) *Task 2*. The shaded region depicts the standard deviation.

### 5.2.2 Deep relational RL

We evaluate the RePreL framework with DRRL as base learners on two domains. The relational state is represented as a *fully-connected graph* in both domains. The first domain is the **Extended Taxi World** (from Chapter 4). The graph representation of a state is obtained as follows. Each passenger in the state is considered a node of the graph, passenger

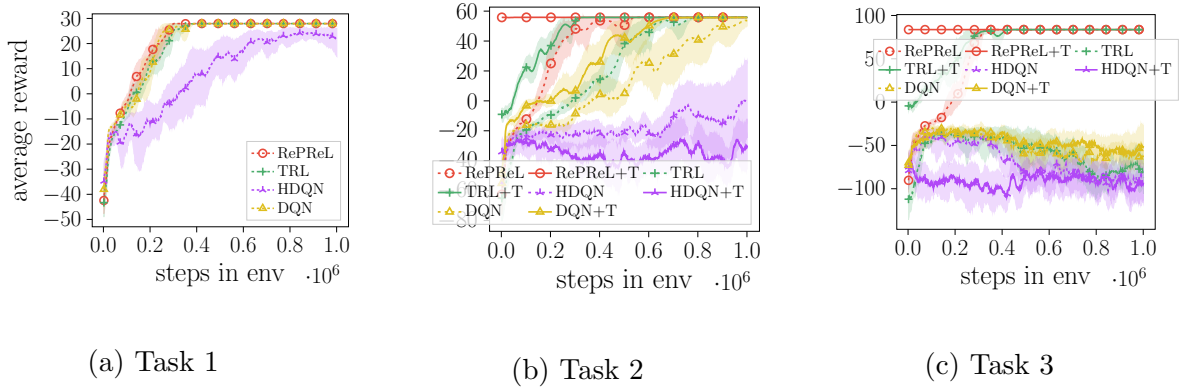


Figure 5.3: Comparing learning curves of deep RL-based learners in the Extended Taxi World. (a) *Task 1* is to drop passenger p1, (b) *Task 2* is to drop p1 and p2, (c) *Task 3* is to drop p1, p2, p3. The shaded region depicts the standard deviation.

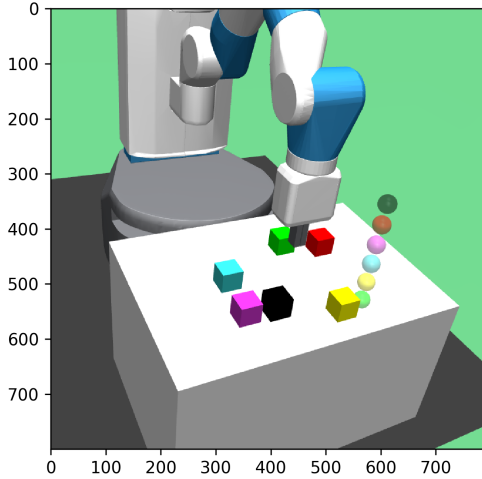


Figure 5.4: FetchBlockConstruction

features are encoded as node attributes, and the taxi location is encoded as a global attribute of the graph. The graph is fully connected.

To demonstrate that RePreL can handle both continuous and discrete spaces (A4) we include a 3-dimensional robotics multi-object manipulation domain with continuous state and action spaces called **FetchBlockConstruction** (Li et al., 2020). FetchBlockConstruction, shown in Figure 5.4 is based on FetchPickAndPlace (Plappert et al., 2018) domain. This domain has multiple blocks and the task is to move each block to its goal location. Each

block is represented as a node in the fully-connected graph with an 18D vector consisting of block position, orientation, relative position, cartesian velocity, angular velocity, and goal location. Features of the robotic gripper (10D vector) are treated as global attributes of the graph. The action space is 4D, consisting of relative change in the 3D position of the two-fingered parallel jaw gripper and the distance between two fingers of the robotic arm. The complete list of D-FOCI statements and the relevant state abstractions are provided in Table 5.3.

We compare the RePReL framework against double DQN in Extended Taxi World. In FetchBlockConstruction, we use a state-of-the-art DRRL method Relational Neural Network (ReNN) by Li et al. (2020). ReNN uses a Soft Actor-Critic (SAC) learner with hindsight experience replay (HER) strategy. In both domains, the network architecture consists of an attention-based message passing module, an attentive graph pooling module, and a multi layer perceptron module. This also demonstrates that the RePReL framework can be adapted to any off-policy RL algorithm, aspect A5. For FetchBlockConstruction we use the same network parameters and experiment settings as Li et al. (2020), except for one change. Instead of using 35 parallel workers, we used 16 workers because of resource constraints. Table 5.2 summarizes the network parameters for the Extended Taxi World.

Figure 5.5 presents the comparison of the RePReL with GNN based DQN agent. Both the agents have comparable performance in *Task 1*, but RePReL shows a significant advantage over DQN in the remaining tasks (A1 and A3). The transferred DQN agents took longer to converge than learning DQN from scratch on *Task 2* and *3*. This might be explained by the mechanism called *capacity loss*, whereby networks trained to predict a sequence of target values lose their ability to quickly fit new functions over time (Lyle et al., 2022; Igl et al., 2021; Ash and Adams, 2020). Figure 5.6a presents the comparison of learning curves of the RePReL agent and the ReNN agent on the task of moving a block to its goal location. While RePReL and ReNN have comparable performance on this task, we present

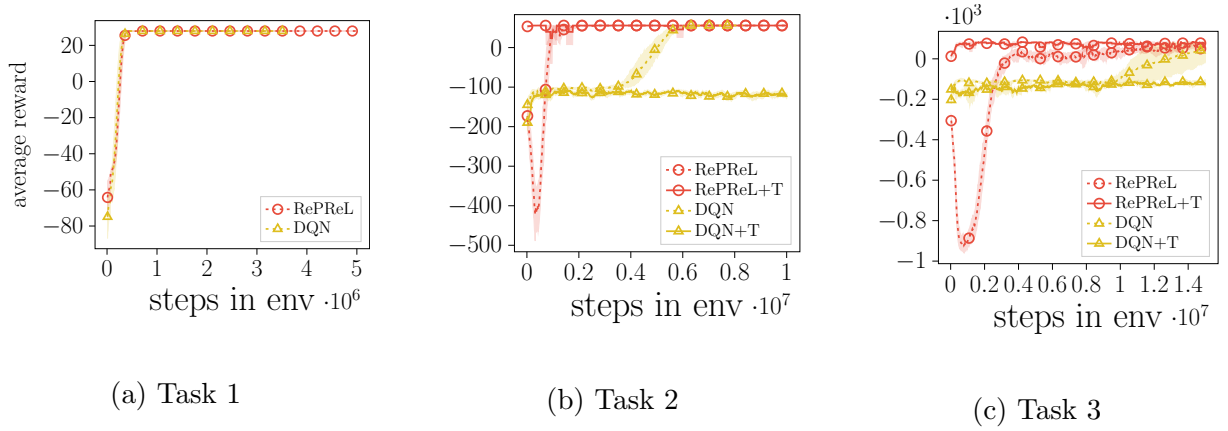


Figure 5.5: Comparing learning curves of deep relational RL-based learners in the Extended Taxi World. (a) *Task 1* is to drop passenger p1, (b) *Task 2* is to drop p1 and p2, (c) *Task 3* is to drop p1, p2, and p3. The shaded region depicts the standard deviation.



Figure 5.6: Comparing deep relational RL-based learners in FetchBlockConstruction. (a) Compares learning curve on the task of moving one block (b) Evaluates generalization for moving 1–4 blocks.

the generalization results to 2–4 blocks in Figure 5.6b. As can be seen, the RePreL agent can easily generalize to multiple objects while being statistically significantly better than the baselines (demonstrating aspect A3).

### 5.3 Conclusion

We extended RePReL for deep, neural, and relational RL agents. Collectively, the empirical evaluations clearly demonstrate our central hypothesis that **RePReL allows for better generalization while exploiting effective abstractions for efficient learning**. The results further demonstrate another important aspect of this formalism—the ability to learn in continuous as well as discrete structured domains. Learning in this setting is generally considered a hard task, and our work takes a small step toward this goal. This work assumes states of the MDP are structured and can be represented as a predicate or graph. In the following chapter, we relax this assumption and allow structured as well as unstructured representation of the state.

Table 5.1: Summary of hyperparameters used in deep RL experiments (Section 5.2.1).

Hyperparameters	Values
Learning rate	0.003
Batch size	128
Max steps	1e6
Max buffer size	1e5
Hidden layers	2
Hidden units	256
Discount rate	0.99
Intrinsic reward on subgoals	30
Office World	
Input size (baseline)	11
Input size (RePReL)	4 pickup & 5 drop
Output size (# of Actions)	4
Output size (metacontroller)	3
Max episode length	1000
Epsilon decay	True
Extended Taxi World	
Input size (baseline)	91
Input size (RePReL )	69
Output Size (# of Actions)	6
Output Size (metacontroller)	2
Max episode length (Task 1)	150
Max episode length (Task 2)	200
Max episode length (Task 3)	300
Epsilon decay	false

Table 5.2: Summary of hyperparameters used in deep relational RL experiment of Extended Taxi World (Section 5.2.2).

Hyperparameters	Values
Number of graph layers	2
Attention embedding	125
Number of attention heads	3
Number of MLP layers	2
MLP embedding	256
Activation function	Leaky ReLU
Batch size	128
Task 1 max episode length	500
Task 2 max episode length	1000
Task 3 max episode length	1500
Epsilon decay	True
Learning rate	0.003
Buffer size	1e6

Table 5.3: D-FOCI statements and relevant features (literals) of the state that form the abstract state.

D-FOCI statements	Option: Abstract state
<b>Office World</b>	
$\{\text{agent-at(L1), move(Dir)}\} \xrightarrow{+1} \text{agent-at(L2)}$ $\{\text{agent-at(L1), move(Dir)}\} \longrightarrow R$ $\text{pickup(X): } \{\text{agent-at(L1), at(X, L), with-agent(X)}\} \xrightarrow{+1} \text{with-agent(X)}$ $\text{pickup(X): with-agent(X)} \longrightarrow R_o$ $\text{deliver(X): } \{\text{agent-at(L1), with-agent(X), office(L), delivered(X)}\} \xrightarrow{+1} \text{delivered(X)}$ $\text{deliver(X): delivered(X)} \longrightarrow R_o$	$\text{pickup(X): } \{\text{agent-at(L1), at(X, L), with-agent(X), move(Dir)}\}$ <hr/> $\text{deliver(X): } \{\text{agent-at(L1), with-agent(X), office(L), move(Dir), delivered(X)}\}$
<b>Extended Taxi World</b>	
$\{\text{taxi-at(L1), move(Dir)}\} \xrightarrow{+1} \text{taxi-at(L2)}$ $\{\text{taxi-at(L1), move(Dir)}\} \longrightarrow R$ $\text{pickup(P): } \{\text{taxi-at(L1), at(P, L), in-taxi(P)}\} \xrightarrow{+1} \text{in-taxi(P)}$ $\text{pickup(P): in-taxi(P)} \longrightarrow R_o$ $\text{drop(P): } \{\text{taxi-at(L1), in-taxi(P), dest(P,L), at-dest(P)}\} \xrightarrow{+1} \text{at-dest(P)}$ $\text{drop(P): at-dest(P)} \longrightarrow R_o$	$\text{pickup(P): } \{\text{taxi-at(L1), at(P,L), in-taxi(P), move(Dir)}\}$ <hr/> $\text{drop(P): } \{\text{taxi-at(L1), in-taxi(P), dest(P,L), at-dest(P), move(Dir)}\}$
<b>FetchBlockConstruction</b>	
$\{\text{armfeat1(AF1), ..., armfeat10(AF10), action(A)}\} \xrightarrow{+1} \text{armfeat1(AF1), ...}$ $\{\text{armfeat1(AF1), ..., armfeat10(AF10), action(A)}\} \xrightarrow{+1} \text{armfeat10(AF10), ...}$ $\text{place(X): } \{\text{blockfeat1(X, BF1), ..., blockfeat18(X, BF18)}\} \xrightarrow{+1} \text{blockfeat1(X, BF1), ...}$ $\text{place(X): } \{\text{blockfeat1(X, BF1), ..., blockfeat18(X, BF18)}\} \xrightarrow{+1} \text{blockfeat1(X, BF1), ...}$ $\text{place(X): } \{\text{armfeat1(AF1), ..., armfeat10(AF10), action(A)}\} \xrightarrow{+1} \text{blockfeat1(X, BF1), ...}$ $\text{place(X): } \{\text{armfeat1(AF1), ..., armfeat10(AF10), action(A)}\} \xrightarrow{+1} \text{blockfeat18(X, BF18), ...}$ $\text{place(X): } \{\text{blockfeat1(X, BF1), ..., blockfeat18(X, BF18)}\} \longrightarrow R_o$	$\{\text{armfeat1(AF1), ... armfeat10(AF10), blockfeat1(X, BF1), ... blockfeat18(X, BF18), action(A)}\}$

## CHAPTER 6

### REPREL EXTENSION FOR HYBRID DATA

In the previous chapter, we presented the deep RePreL framework, which is a combination of a high-level symbolic reasoner and lower-level neural reasoners. It assumes that the state representation is structured, that is each state variable is symbolic and has clear semantics. However, in many real-world domains, the state has sub-symbolic elements. Data is heterogeneous, a hybrid of structured and unstructured representation. In this chapter, we introduce a novel neuro-symbolic system, Hybrid Deep RePreL (Kokel et al., 2022) that is able to learn policies in a hybrid environment.

#### 6.1 Introduction

Building a two-level neurosymbolic system that combines a higher-order symbolic reasoner with a lower-level fast deep learner is not only a long cherished dream but is one of the more popular research directions inside AI (Booch et al., 2021). Particularly, exploring the combination of learning and neurosymbolic processing in the context of sequential decision-making is quite natural (Nilsson and Ziemke, 2007; Anderson et al., 2020; Mitchener et al., 2022) given the recent success of DRL methods on large-scale tasks (Silver et al., 2016, 2017, 2018). The RePreL system takes a first step in the direction of combining (relational) planning and RL in solving structured problems by using the planner to define a smaller set of (abstract) state-action spaces to allow for efficient learning by the lower level RL agent. The key success of the RePreL method lies in its capability of generalization to a varying number of objects.

While RePreL was successful, it had an important assumption—the underlying state variables are symbolic. The higher-order symbolic planner operated at the level of predicates defined in first-order logic and then constructed appropriate grounded predicates as features

for the RL agent. This assumption restricts the RePReL framework in two specific ways. First is that the power of DRL methods is not effectively exploited, as DRL works best in image or tensor representations. Second, the approach cannot fully exploit the fusion of multi-modal data where the data could arrive from multiple sources—images, text, and potentially a feature-based descriptor.

This chapter relaxes the symbolic feature assumption and extends the RePReL framework to construct a *Hybrid Deep Relational Planning and Reinforcement Learning (HRePReL)* architecture that **fuses** multi-modal information from two or more sources. In this chapter, we make the following contributions.

1. We extend the standard RePReL architecture to handle multi-modal data by fusing information from multiple sources.
2. The resulting HRePReL framework is a neurosymbolic architecture that leverages the deliberative nature of a higher-level planner with the fast learning nature of the lower-level deep network.
3. Finally, our empirical evaluations on two domains clearly demonstrate the effectiveness of learning and the efficiency of generalization and transfer across related tasks.

The rest of the chapter is organized as follows. Section 6.2 presents the HRePReL architecture, Section 6.3 present our empirical evaluations, and Section 6.4 concludes the chapter by presenting the directions for future research.

## 6.2 Hybrid deep RePReL

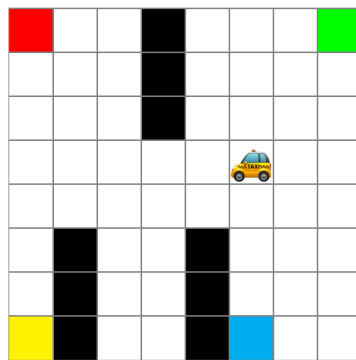
To address multi-task hybrid RL problems, with structured and unstructured data, there is a need to incorporate data from different sources. For convenience, the proposed approach assumes the unstructured data is an image, but it can be extended to other forms of unstructured data verbatim.

**Given:** A combination of structured (object descriptions as predicate logic representations) and unstructured (images or text) data.

**To Do:** Develop a hybrid architecture that learns to act.

Consider the scenario of a ride-sharing app, the information required for booking and riding a passenger arrives from different sources. The local geography of the region is obtained from a map, the location of the cab can be obtained from the cab driver's mobile, and the location of the rider and the destination of the ride are obtained from the rider's mobile. In such cases, the information required for a ride can be a conglomeration of many varied types of data coming from different sources.

As a more concrete example, consider a hybrid taxi domain, shown in Figure 6.1. There are two passengers in this domain and one taxi. Six actions available in this domain are: **east, west, north, south, pick, drop**. The task is to transport passenger(s) from their current location to their destination location. Only *one* passenger can hire the taxi at a time. We consider **three different sources of information** in this domain, the taxi location and the geography of the region are available as an image from one source (see Figure



(a)

$at(p1, l1), dest(p1, d1)$

$at(p2, l2), dest(p1, d2)$

(b)

Figure 6.1: (a) Taxi location and geography information available as an image. (b) Passenger location and destination information as state predicates from a passenger's mobile.

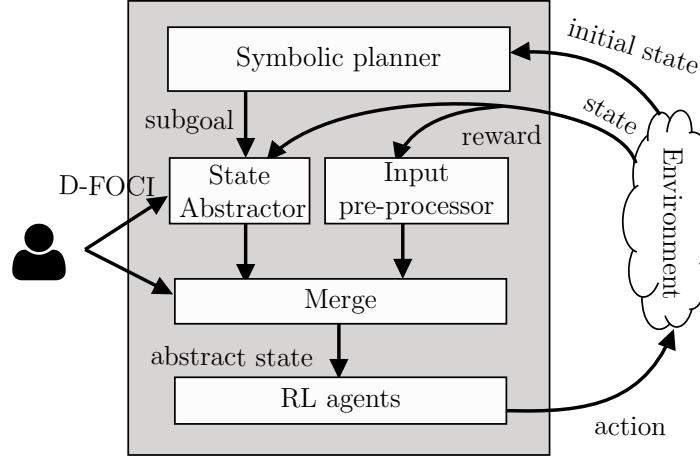


Figure 6.2: Proposed HDRePreL architecture.

6.1a); the current location and the destination location of the passenger  $p_1$  are available from  $p_1$ 's mobile as state predicates; and similarly the current location and the destination of the passenger  $p_2$  is available from  $p_2$ 's mobile (see Figure 6.1b). Hence, the state or the observation from the environment is hybrid—consisting of the structured data from passengers' mobile and unstructured image data with taxi location.

For such hybrid RL domains, we extend the RePreL framework and introduce hybrid deep RePreL (HDRePreL). Figure 6.2 presents the proposed HDRePreL architecture. The HDRePreL differs from the previous deep RePreL framework in the following ways.

1. We extend the D-FOCI language in HDRePreL to allow for latent predicates.
2. We add two extra modules, an input preprocessing module and a merge module, to handle the combination of structured and unstructured information.

The symbolic planner and the low-level reinforcement learner remain the same. Next, we explain each of these differences in greater detail.

## D-FOCI extension

D-FOCI language (see §4.3.4) consist of statements of the form,

$$[\langle \text{option} \rangle] : \langle \text{influent} \rangle \xrightarrow{[+1]} \langle \text{resultant} \rangle \quad (6.1)$$

where **influent**s is a finite set of literals, **resultant** is a single literal, and the **option** is a temporally extended operator from the symbolic planner. Additionally, action variables are allowed in the **influent** and reward variables are allowed in **resultant**. D-FOCI statement states that when executing the given **option**, the **resultant** literal in time-step  $t + 1$  is influenced by literals in **influent**s in time-step  $t$ .

Inspired by the work of Manhaeve et al. (2018), we extend the first-order language used by D-FOCI to include latent predicates for hybrid, and heterogeneous domains. Latent predicate literals are allowed in the **influent**s as well as the **resultant**.

For example, in the hybrid taxi domain, the location of the taxi is available from the image so a latent predicate is introduced to represent the taxi location—**Img:taxi\_at**. The passenger location and destination are available as state predicates, so they are represented using the standard first-order logic notation. The location of the taxi is influenced by its previous location and the action performed, which is captured in the D-FOCI statement in Equation 6.2a. Further, when executing the task of picking up a passenger, if one assumes that the taxi is empty, then it can be safely inferred that the passenger’s location, the passenger in the taxi, and the taxi’s location are the only influents of the completion of the task. This influence is captured in Equations 6.2b and 6.2c. Similarly, the influence information while dropping passenger  $P$  is captured in Equation 6.2d–6.2f.

$$\{\text{action}, \text{Img:taxi\_at}(X)\} \xrightarrow{+1} \text{Img:taxi\_at}(X) \quad (6.2a)$$

$$\text{pick}(P) : \{\text{action}, \text{in\_taxi}(P), \text{at}(P, Y), \text{Img:taxi\_at}(X)\} \xrightarrow{+1} \text{in\_taxi}(P) \quad (6.2b)$$

$$\text{pick}(P) : \{\text{in\_taxi}(P)\} \longrightarrow \text{Reward} \quad (6.2c)$$

$$\text{drop}(P) : \{\text{at\_dest}(P)\} \longrightarrow \text{Reward} \quad (6.2d)$$

$$\text{drop}(P) : \{\text{at}(P, X), \text{dest}(P, D), \text{at\_dest}(P)\} \longrightarrow \text{at\_dest}(P) \quad (6.2e)$$

$$\text{drop}(P) : \{\text{action}, \text{Img:taxi\_at}(X), \text{at}(P, Y), \text{in\_taxi}(P)\} \xrightarrow{+1} \text{at}(P, K) \quad (6.2f)$$

The process of identifying the abstract state representations using the D-FOCI statements with latent predicates remains the same as presented in Section 4.3.6. Note that the extracted abstract state might have latent predicates.

### Input Pre-processor

This module processes the unstructured part of the state observation and provides latent state embeddings to the *merge* module. The input pre-processing module can be a Convolutional Neural Network (CNN) for image data, a transformer for text data, or a combination of both (we employ CNNs in our experiments). The neural network of the input pre-processor can be chosen based on the type of unstructured data.

### Merge module

The relevant state variables obtained from the state abstractor and the latent predicates obtained from the input pre-processor serve as inputs to the merge module. The relevant state predicates from the state abstractor would include the latent predicates. The merge module would replace the latent predicates with the respective latent embedding using the D-FOCI statements. The derived abstract state then serves as the input to the RL agent.

## HDRPreL Learning

HDRPreL learning procedure is the same as the deep RePreL batch learning (see §5.1.1). However, instead of just updating the RL agent network while training, in HDRPreL we update the input preprocessor as well as the RL agent network.

### 6.3 Experiments

We evaluate HDRPreL in two domains, Taxi and Craft World. We design our experiments to explicitly answer the following questions.

**Q1. Sample Efficiency:** Do the abstractions induced in HDRPreL improve sample efficiency?

**Q2. Generalization:** Does HDRPreL efficiently generalize to varying number of objects?

#### 6.3.1 Domains

The first domain that we consider is the **hybrid taxi domain**. This is an  $8 \times 8$  grid, shown in Figure 6.1a, with one taxi and 4 special locations:  $\langle R, G, B, Y \rangle$ . There can be more than

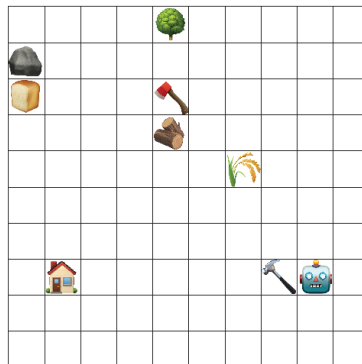


Figure 6.3: Craftworld domain.

one passenger at a time in the grid, but the taxi can be hired only by a single passenger. In every episode, the pickup and drop locations of each passenger are sampled from the 4 special locations. The agent can perform 6 actions: *up*, *down*, *right*, *left*, *pick*, *drop*. The taxi location and the grid are provided as a grayscale image and the passenger information is provided as a vector. The environment provides a reward of  $-0.1$  for every step and  $-1$  for pick or drop action in wrong locations. We consider the following three tasks in this domain: *task 1*, drop a passenger to their destination; *task 2*, drop two passengers to the respective destinations; and *task 3*, drop three passengers to their destination. The D-FOCI statements used in this domain are presented in Equation 6.2.

The second domain is a Minecraft-inspired grid-world domain called Craftworld (Devin et al., 2019). This environment contains a 10x10 grid, as shown in Figure 6.3. This domain has 7 different objects, some of which are holdable. The six actions available in this domain are the same as that of the hybrid Taxi domain. In this domain, the agent’s location and the map are available as unstructured data—as an image, whereas all the objects’ locations are available as state predicates. We consider three tasks in this domain: *Task 1*, make bread by bringing an axe to the wheat’s location; *Task 2*, build a house by bringing a hammer to the wood’s location; *Task 3*, break a rock by bringing a hammer to the rock’s location. The set of D-FOCI statements used in this domain is presented in Equation 6.3.

### 6.3.2 Baselines

We evaluate our HDRePreL agent against a double *Deep Q-Network (DQN)* (van Hasselt et al., 2016), a state-of-the-art DRL method. The DQN agent learns a single end-to-end policy for completing the task. The network architecture of the DQN agent includes a CNN module—equivalent to the input pre-processor. The CNN module receives the unstructured part of the state observation. Then, the output of the CNN module is concatenated with the structured part of the state representation and passed through a multi-layered perceptron.

The network parameters are summarized in Table 6.1. For honest comparison, we employed the same network parameters in HDRePReL and DQN.

### 6.3.3 Sample efficiency

To evaluate the efficiency of the HDRePReL architecture, we compare it against the DQN agent. We compare the learning curves of these two agents on three tasks of the hybrid taxi domain in Figure 6.4 and on the three tasks of the craft world in Figure 6.5. Ignore the dashed lines with ‘+T’ for now. While both the HDRePReL and the DQN agent achieve the optimal reward after 200K steps in Taxi *task 1*; their performance significantly differs in the five remaining tasks. In these tasks, it can be clearly seen that the efficiency of HDRePReL is significantly better than the baseline thus demonstrating that the abstractions defined by the reasoner enable efficient learning. Hence, we answer Q1 affirmatively in that HDRePReL significantly improves the *sample efficiency* when compared to an end-to-end RL agent that does not use any domain-specific knowledge.

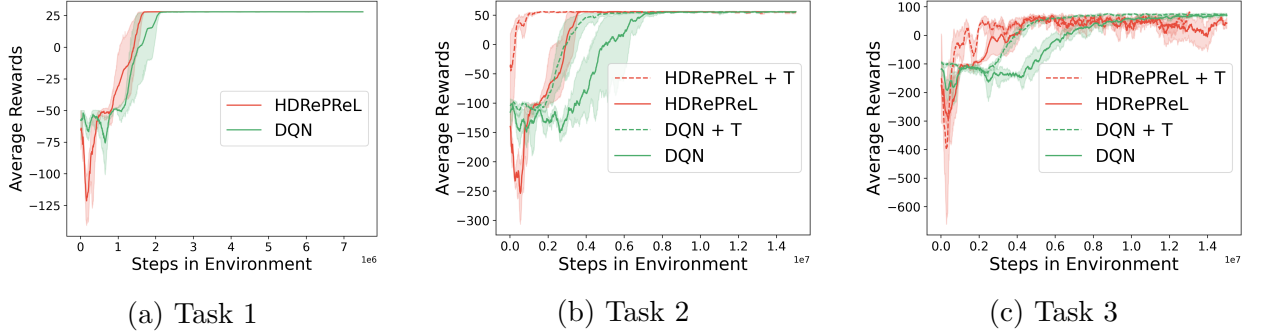


Figure 6.4: Comparing learning curves of hybrid deep RePReL with DQN in Hybrid Taxi World. (a) *Task 1* is to drop passenger p1, (b) *Task 2* is to drop p1 and p2, (c) *Task 3* is to drop p1, p2, p3.

### 6.3.4 Generalization

We evaluate the generalization capability of the HDRePReL agent in the hybrid taxi domain. We transfer the HDRePReL and the DQN agent trained on *task 1* and train them

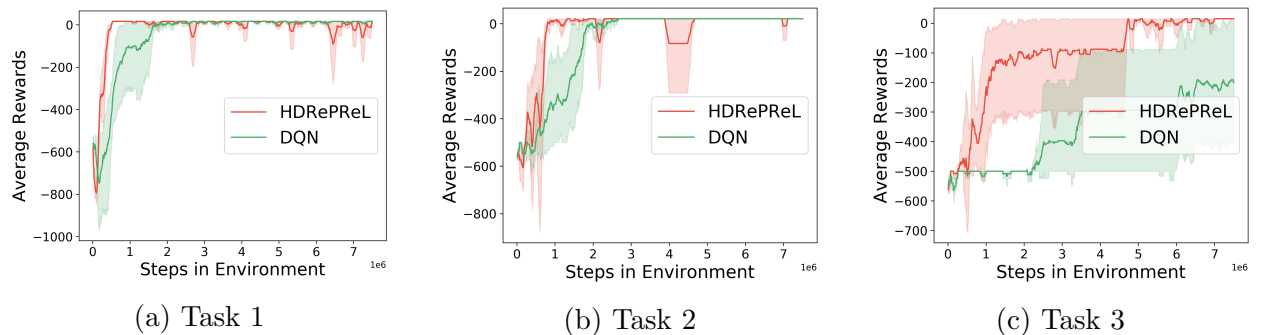


Figure 6.5: Comparing learning curves of HDRePreL with DQN in Craft World. (a) *task 1* is to make bread; (b) *task 2* is to build house; (c) *task 3* is to break rock.

on *task 2*. Subsequently, we transfer the agents from *task 2* to *task 3*. Figures 6.4b and 6.4c present the learning curves of these transferred agents indicated by ‘+T’. While both the transferred agents have steeper learning curves than their respective base models, the transferred RePreL agent converges significantly faster than the transferred DQN agent. In many of the tasks, this is achieved with no learning in the new domain. This is due to the inherent generalization capabilities of the HDRePreL agent. These transfer results in the hybrid taxi domain allow us to answer Q2 affirmatively in that HDRePreL allows for successful generalization across a varying number of objects and is best suited for relational domains.

## 6.4 Conclusion

To summarize, the experiments conclusively demonstrate the most important observation about the HDRePreL agent – that it leverages the generalization power of the symbolic planner with the efficient learning ability of the underlying DRL agent. The resulting combination is a powerful neurosymbolic system that not only learns efficiently but generalizes to a larger number of objects by bootstrapping on its prior learned policies. The generalization ability is particularly important when learning occurs with different starting states, differing numbers of objects, different domain configurations, or different target states. In

the real world, assuming that the data arrives only from a single source can lead to disastrous results. Hence, using architectures that support heterogeneous data that have the ability to generalize to different numbers of objects is crucial for a system to be deployed in real-time. HDRePReL takes a step in this direction.

This chapter presented a novel neurosymbolic system that is capable of learning in the presence of heterogeneous (discrete and continuous), hybrid (structured and unstructured), and relational (objects and relations) data. HDRePReL combines the advantage of a deliberate relational planner with a fast DRL agent. The resulting combination demonstrated both effective learning and efficient generalization across different numbers of objects. More rigorous evaluation of the system on larger problems is an immediate future direction. Allowing for the DRL agent to communicate back to the planner in order to refine the planner based on new and interesting observations is a high-impact direction that could allow for a fully differentiable end-to-end system. Finally, given the use of a symbolic planner, the resulting decompositions and abstractions are explainable. Thus allowing for richer human interaction with the given system remains an interesting direction for future research.

$$\begin{aligned}
&\{\text{action}, \text{Img:agent\_at}(X), \text{holdable}(Y), \text{holding}(Y)\} \xrightarrow{+1} \text{Img:agent\_at}(X) \\
&\{\text{holdable}(Y), \text{holding}(Y), \text{at}(Y, L)\} \xrightarrow{+1} \text{holding}(Y) \\
&\{\text{at}(\text{rock}, L1), \text{at}(\text{tree}, L2)\} \xrightarrow{+1} \text{Img:agent\_at}(X) \tag{6.3} \\
&\{\text{at}(\text{tree}, L), \text{Img:agent\_at}(L), \text{holding}(\text{hammer})\} \longrightarrow \text{at}(\text{tree}, L), \text{at}(\text{wood}, L) \\
&\{\text{at}(\text{wheat}, L), \text{Img:agent\_at}(L), \text{holding}(\text{axe})\} \longrightarrow \text{at}(\text{wheat}, L), \text{at}(\text{bread}, L) \\
&\{\text{at}(\text{wood}, L), \text{Img:agent\_at}(L), \text{holding}(\text{hammer})\} \longrightarrow \text{at}(\text{wood}, L), \text{at}(\text{house}, L) \\
&\{\text{at}(\text{rock}, L), \text{Img:agent\_at}(L), \text{holding}(\text{hammer})\} \longrightarrow \text{at}(\text{rock}, L) \\
&\text{pick}(P) : \text{holding}(P) \longrightarrow R_o \\
&\text{go\_to}(P) : \{\text{at}(P, L), \text{Img:agent\_at}(L)\} \longrightarrow R_o
\end{aligned}$$

Table 6.1: Summary of the network hyperparameters

Hyperparameters	Values
Learning rate	0.003
Batch size	128
Max steps	1e6
Max buffer size	1e5
Discount rate	0.99
Intrinsic reward on subgoals	30
Number of CNN Layers	2
CNN Kernel Size	4
CNN Stride	1
CNN Activation Function	<i>relu</i>
Epsilon decay	True
Output Size (# of Actions)	6
RL Hidden layers	2
RL Hidden units	256
Craftworld	
Image size	10x10x1
Structured Input Size	48
Max episode length	500
Hybrid Taxi Domain	
Image Size	8x8x1
Structured Input Size	27
Max episode length (Task 1)	500
Max episode length (Task 2)	1000
Max episode length (Task 3)	1000

## **PART III**

### **COLLABORATIVE PROBLEM SOLVING**

## CHAPTER 7

### PLANNING AND LEARNING VIA COMMUNICATION

In this chapter, we consider the problem of human-machine collaboration in the context of a collaborative building task in Minecraft. To this effect, we present an integrated system (Lara) that builds on advancements in several related fields - NLP, knowledge representation, inductive logic programming, planning, and statistical relational AI. Specifically, Lara consists of a language parser and generator for effective communication, a rich representation based on first-order logic that allows for generalization, a concept learner that is capable of generalizing from a small number of instances by effectively exploiting human guidance and a planner capable of exploiting domain knowledge effectively. The resulting integrated system is presented and demonstrated in detail here.

#### 7.1 Introduction

It is well known that human-machine collaborative planning and problem-solving are quite challenging as it requires a shared perception of the world, sophisticated language understanding, glitch-free execution, bi-directional communication, and contextual understanding. Specifically, we consider the task of collaborative building in Minecraft (Kokel et al., 2021, 2022) and develop *an integrated system* that builds on several different areas – hierarchical planning (Bercher et al., 2019; Erol et al., 1994; Nau et al., 1999), knowledge representation and reasoning (Brachman and Levesque, 2004), inductive logic programming (Cropper and Dumancic, 2022; Muggleton and Raedt, 1994; Raedt and Kersting, 2008), knowledge-based learning (Towell and Shavlik, 1994; Kokel et al., 2020), natural language processing (Banarescu et al., 2013; Bahdanau et al., 2015; Sutskever et al., 2014) and generation (Gatt and Krahmer, 2018), and statistical relational AI (Raedt et al., 2016).

It is natural to focus on the modality of communication such as gestures, or natural language when building human-AI collaborative systems. However, it is also essential to

establish a common vocabulary for communication. This is especially important in a complex domain such as Minecraft, where given the basic definitions of elementary shapes and sizes, higher-order concepts should be built. The key requirement is that the common vocabulary of concepts keeps growing as more tasks are solved and the interactions increase. The set of concepts should be easily learnable (with a small number of examples) and generalizable. To this effect, we assume the existence of a basic vocabulary and build upon an *inductive logic programming based concept learner* (Das et al., 2020). This concept learner learns a set of hierarchical concepts based on a very small (possibly one) number of examples using domain knowledge as an inductive bias.

While learning these generalized concepts, there is a necessity for the system to continue interacting in the environment and modifying its interactions based on the induced concepts and the feedback both from the environment and the human. The induced concept must be both generalizable and compositional. Generalizable to different dimensions, sizes, and colors and compositional so as to effectively employ the hierarchies of concepts that are induced by the concept learner. The induced concepts are used as preconditions to guide a hierarchical task planner (Das et al., 2018) in our framework. The planner uses domain knowledge and actively seeks human guidance in the form of knowledge constraints that are then used for both efficient and effective planning.

Finally, for effective communication, both the modality and the representation need to be established. For modality, we employ the use of NLP parsers (both rule-based and neural-based parsers) to translate the commands from the humans in natural language to a formal internal representation. And for the reverse communication from the internal representation to natural language, we restrict ourselves to specific forms and templates. Extending this to allow for richer neural language generators is our envisioned future work. For the internal representation, we employ the use of abstract meaning representations (Banarescu et al., 2013) that allow for capturing generalized knowledge.

We make the following key contributions: (1) we present an integrated system called Lara (Planning and learning via communication), that obtains instructions and knowledge in rich natural language, reasons with the observations and knowledge, and executes the plan automatically; (2) the system is capable of obtaining human “advice” as constraints both to learn the hierarchical concepts and to perform planning; (3) most importantly, the system is capable of soliciting this advice in an active manner, thus reducing the effort needed from the human in collaborative planning and execution.

The remaining chapter is organized as follows. Section 7.2 introduces the necessary background of the components in the system. Section 7.3 presents the overall system design with example scenarios. Section 7.4 demonstrates the system. Section 7.5 reviews the related work. Finally, Section 7.6 concludes the chapter by discussing the salient features of the system and outlining areas of future research.

## 7.2 Preliminaries

### 7.2.1 Concept learning

Inductive logic programming (ILP) learns rules or relations inductively from the given set of background knowledge and positive as well as negative examples (Muggleton and Raedt, 1994). The rules or relations in ILP are learned as declarative logic programs, often as horn clauses. *Concept learning* in ILP, essentially, reduces to learning a clausal theory (a first-order logic program) that covers as many positive examples of the target concept as possible and as few negative examples as possible (Raedt, 1997). ILP employs background knowledge as a search bias to constrain the space of the hypothesis. Golem (Muggleton and Feng, 1990), FOIL (Quinlan, 1990), Progol (Muggleton, 1995), TILDE (Blockeel and Raedt, 1998), Aleph (Srinivasan, 1999), FOCL (Pazzani et al., 1991) are some examples of ILP systems that learn the target concept by induction. One important aspect of a

cogent concept learning framework is that it should represent concept hierarchies, allowing us to induce more complex concepts given previously learned ones, as opposed to learning traditional logic programs with one level of abstraction (Fu and Buchanan, 1985)

### **7.2.2 Neural parsers**

Abstract Meaning Representation (AMR) is a semantic representation of natural language (Banarescu et al., 2013), where concepts are represented as nodes and relations between concepts are represented with edges. AMR can be systematically translated to first-order logic (Bos, 2016) and, hence, is a suitable representation for integration with ILP systems. With the tremendous success of neural networks to solve various sequence-to-sequence problems (Bahdanau et al., 2015; Sutskever et al., 2014), multiple neural models are proposed for parsing text to AMRs (Xu et al., 2020; Konstas et al., 2017; Zhang et al., 2019; Peng et al., 2017).

### **7.2.3 Minecraft**

Minecraft is a popular computer-based game developed by Mojang Studios and released in 2011, <https://minecraft.net>. Minecraft exposes a virtual 3D world where player avatars can explore the world, travel on adventures, build structures, hunt for food, harvest raw materials, craft tools, and kill zombies. It poses many challenging problems which have intrigued researchers to use it as a test bed for various open problems in AI (Kanervisto et al., 2021; Salge et al., 2020; Shah et al., 2021). Project Malmö (Johnson et al., 2016), built on top of Minecraft, exposes an API for flexible AI experiments.

## **7.3 Lara - Planning and learning via communication**

We now present the details of our collaborative problem-solving system and discuss each of the components in detail.

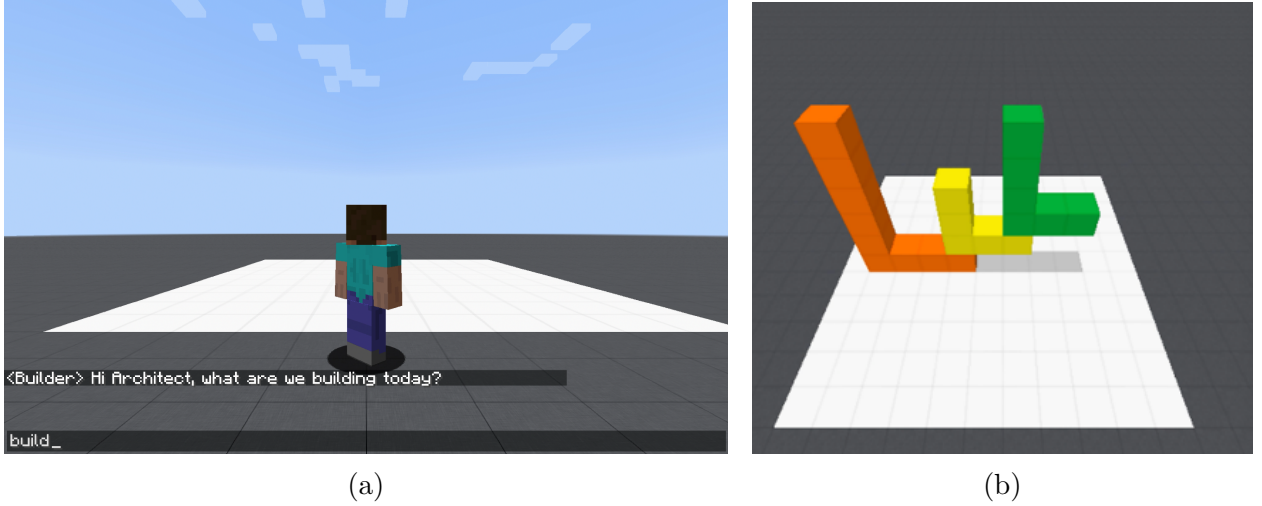


Figure 7.1: (a) Minecraft builder screen showing the 3D build region and the chat interface. (b) Example of a target structure in the oracle screen. The architect can see both screens.

### 7.3.1 Problem definition

This work considers the problem of human-machine collaboration in a Minecraft environment. A *collaborative building task* (Jayannavar et al., 2020; Kokel et al., 2021; Narayan-Chen et al., 2019; Narayan-Chen, 2020) is defined in the context of Minecraft where a human and a machine have to collaborate to build a target structure by block placements. Blocks are restricted to be one of six colors: red, blue, green, purple, orange, and yellow. A fixed 3D grid of size  $11 \times 9 \times 11$  is defined as a build region, as shown in Figure 7.1a. The avatar can only move and place blocks in this build region. The task is to build a target structure in this stipulated build region. The target structure candidate is sampled from a preset list of complex shapes and displayed in a separate oracle window. An example of a target structure is shown in Figure 7.1b. Two players, an architect and a builder, collaborate and communicate using natural language via the chat interface. The architect and the builder take turns on the chat window.

The architect can view the target structure in the oracle window and can also see the current state of the build region. The builder can not see the oracle window. It can move in

the build region to place and remove blocks. The role of the architect is played by humans and the role of the builder is played by Lara. The game begins with a simple target structure in the oracle window and a greeting from the builder Lara to the architect (as seen in Figure 7.1a).

For a successful target structure construction and most importantly, generalization of the learned concepts, the architect must decompose the target structure into smaller structures and instruct Lara to achieve those subtasks. Lara must parse the instructions, seek clarifications as appropriate, and execute the subtask(s). A sample interaction between an architect and the builder Lara is shown in Figure 7.2, where the target structure is a red *L*. The challenges posed by the Minecraft-based blocks world task are as follows:

1. The communication between the architect and the builder is inherently bi-directional (see for example Figure 7.2).
2. The builder should be able to seek clarifications as required.
3. Both players must share some initial structures in the vocabulary, expand the vocabulary with experience, agree upon the changes and reuse the learned higher-level concepts as appropriate. This requires an effective reasoning process over the learned concepts.

These problems of the proposed task highlight the key challenges of the collaborative planning problem: *bi-directional communication, contextual understanding, composable vocabulary, and a powerful concept learner that can induce new, rich concepts based on limited interaction and experience.*

**Our key contribution in this work is the demonstration of our collaborative planning and problem-solving agent** that addresses these key challenges. Our system has the capability to understand, quantify and measure “what-it-doesn’t-know” (dearth

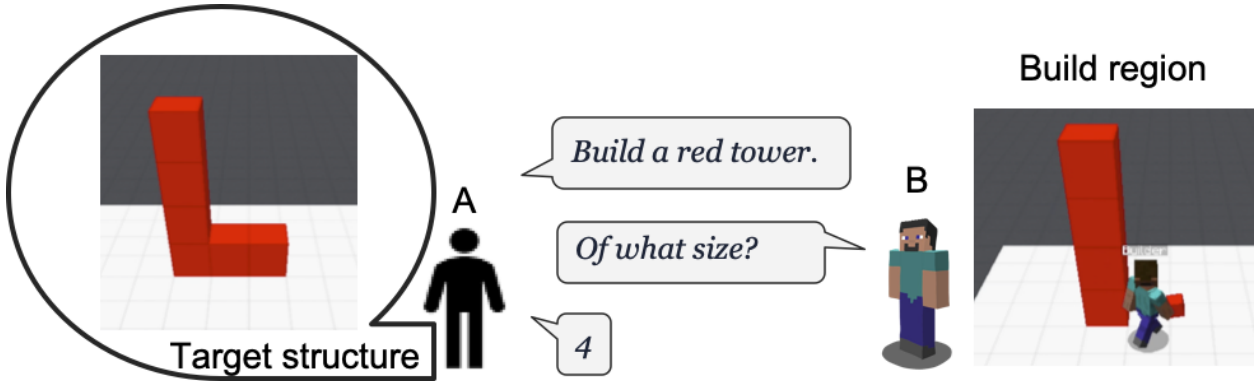


Figure 7.2: Target structure on the left is visible only to the architect (A). The architect instructs the builder (B) to build a red tower. B seeks clarification about the size and then proceeds to build the tower in the build region.

of relevant information) and leverage that understanding to elicit “advice/knowledge/constraints” at the most appropriate decision points from the humans and potentially learn better plans for increasingly complex structures. Some recent works (for e.g. Narayan-Chen et al. 2019 and Köhn et al. 2020) introduced a similar Minecraft environment, but focused on the dialogue generation and instruction giving; instead of the dialogue understanding, concept induction, and planning challenges we focus on here.

### 7.3.2 System setup

For the collaborative building task in Minecraft, a few essential pieces of prior knowledge (domain information) are assumed to exist in both the builder and the architect. These essentials include directions, primitive structures, block indicators, and six colors. For simplicity, we only use the directions w.r.t the architect’s viewpoint. Eight primitive structures include a block, tower, row, column, cube, cuboid, square, and rectangle. Five of them are shown in Figure 7.3a. Three primitive shapes not shown here include a single block, a square lying on the floor, and a rectangle lying on the floor. Block indicators include guides for pointing to a single block that is a component of the structure, a few examples are shown in Figure 7.3b–7.3d. Additionally, the terms height, width, and length are used to represent

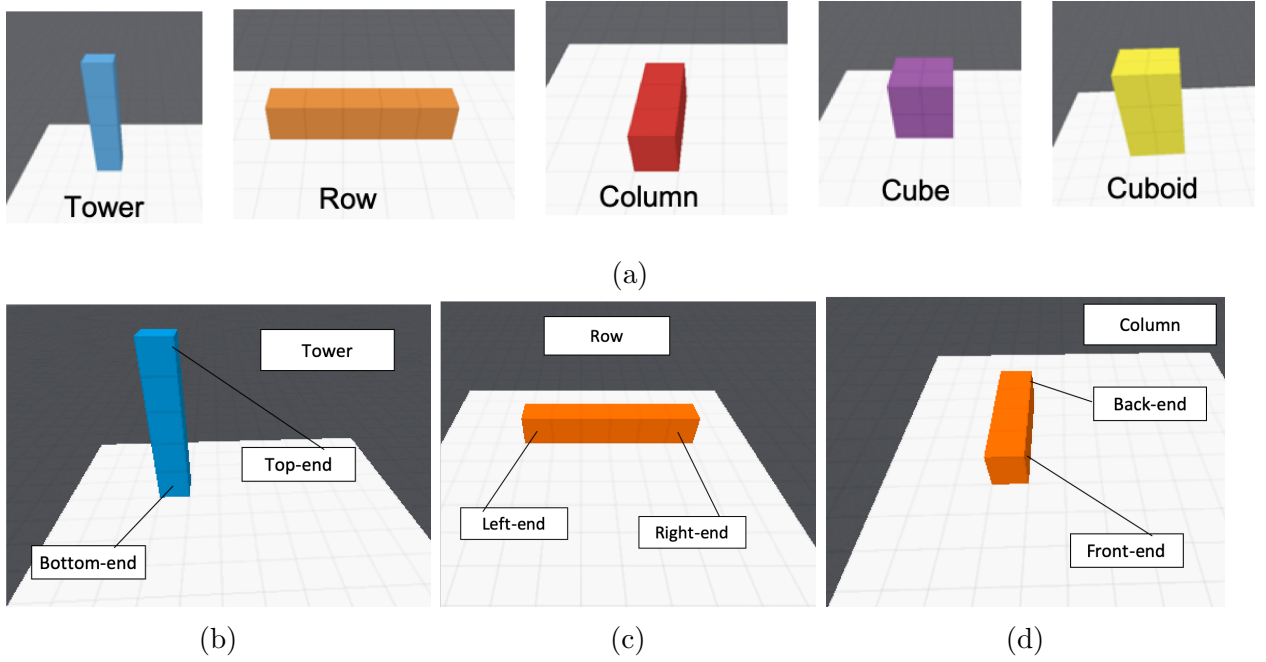


Figure 7.3: (a) Five of the eight primitive structures: tower, row, column, cube, and cuboid. Block indicators that point to a single block of the structure. (b) top-end and bottom-end, (c) left-end and right-end, (d) back-end and front-end. For complex structures, these indicators can be combined, for example, the front-bottom-left block of a cube.

the size of the structure from top-end to bottom-end, left-end to right-end, and front-end to back-end, respectively. So, the tower size is its height, the row size is its width, and the column size is its length. These elements of initial knowledge primitives of the domain are later expanded upon by the learner.

With this initial knowledge, the architect instructs the builder to build the target structure. In our system, the architect can either write a natural language text to instruct the builder or write “UNDO” to revoke the last instruction. Upon receiving the instruction, the builder can either execute the instruction or seek additional information for clarification (see for e.g. Fig. 7.2), provide prompts so that the architect can provide instruction in a fashion that is comprehensible to the builder. Once the target structure is constructed, the architect states “done”. This would instruct the builder that the task of building the target structure is accomplished in the build region. When the structure building is completed, the builder

offers to remember the structure for reuse. The architect can then provide a name for the structure and describe its height, width, and length to the builder. The builder might ask some yes/no questions to induce a generalized concept of the structure (Das et al., 2020). If the builder is successful, the new structure is then added to the builder’s capabilities and can then be treated as a lower-level structure. The process continues so that more complex concepts are introduced as needed.

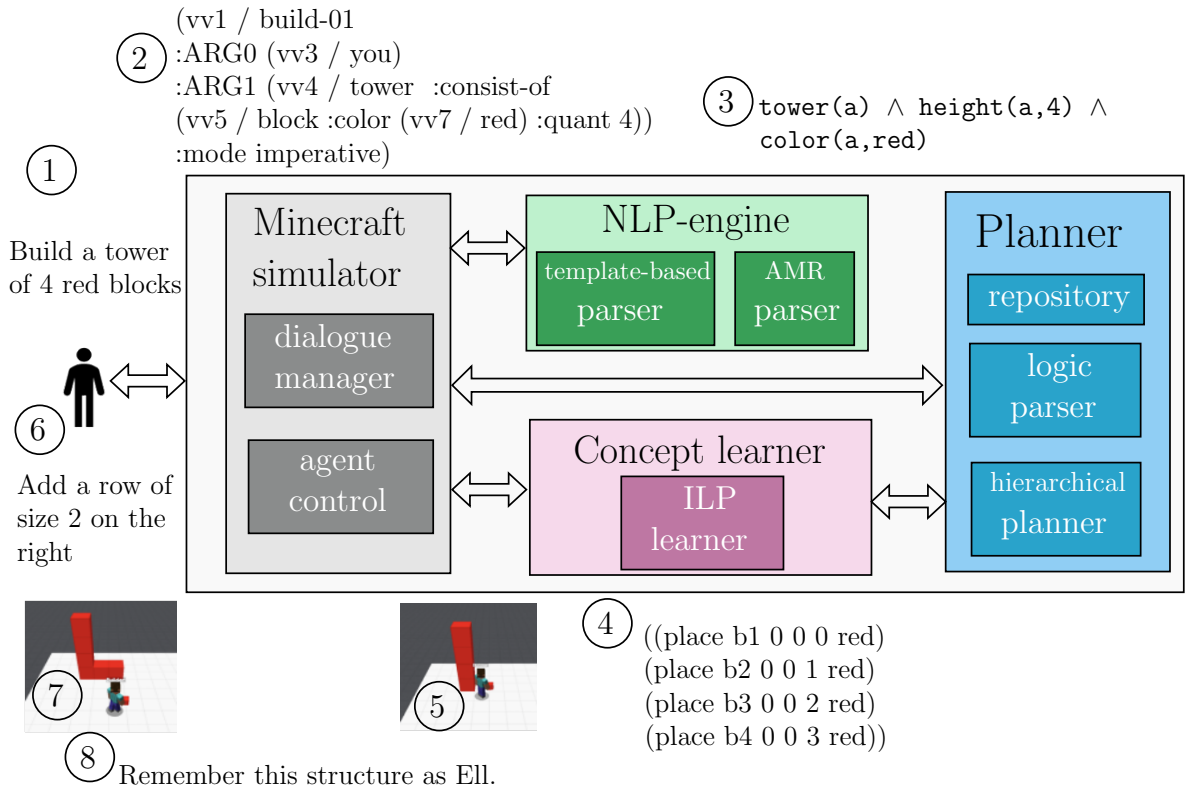


Figure 7.4: Architecture of Lara. It consists of a Minecraft simulator, NLP engine, planner, and concept learner. An example flow of building the “L” shape is illustrated. ① Natural language instruction by the architect to build a red tower. ② AMR representation of the instruction parsed by AMR parser. ③ Logic representation of the instruction. ④ Block placement plan generated by the planner. ⑤ Agent control executes the plan in the build region. ⑥ Next instruction from the architect to add a row. ⑦ Agent control executes the next instruction in the build region. ⑧ Structure saved as “L”.

### 7.3.3 System architecture

The architecture of our system Lara is illustrated in Figure 7.4 that integrates different research components to develop a human-machine collaborative system. It primarily consists of four key components: *Minecraft simulator*, *NLP engine*, *Planner*, and *Concept learner*. The Minecraft system and the interfacing APIs form the Minecraft simulator. The module processing natural language text forms the NLP engine. The module generating the action plan for the Minecraft avatar constitutes the planner. Finally, the module learning new structures forms the concept learner component. We now describe each of these components in greater detail.

### 7.3.4 Minecraft simulator

Project Malmo (Johnson et al., 2016) is extended for our collaborative building task by adding a dialogue manager and agent control module. The **dialogue manager** has two major responsibilities, 1. to triage the messages from the architect and 2. to generate natural language text for bi-directional communication. A message from the architect could either be a build instruction, a clarification, a concept explanation, or an UNDO operation. Dialogue Manager would accordingly pass forward the request to either the NLP engine, the planner, the concept learner, or the agent-control module, respectively. For bi-directional communication, the dialogue manager maintains a fixed set of template sentences with slots. These slots are then populated accordingly as required and sent via the chat interface. The set of template sentences maintained by the dialogue manager is provided in Appendix C.1. The **agent control** module processes the plan generated by the planner and sends the action commands to MALMO API for execution in the Minecraft environment.

### 7.3.5 NLP engine

The main job of the NLP engine is to parse the natural language instructions from the architect and provide a **generalized semantic representation** of such instructions for processing. We leverage first-order logic (FOL) to encode this semantic representation. We choose this for two specific practical reasons. First, FOL representation is compatible with the ILP-based concept learner. Second, Planning Domain Description Language (PDDL) is also a first-order predicate representation. Given the compatibility of FOL representation with the two components (the concept learner and the planner), it was a natural choice. While the above two reasons are from a pragmatic point of view, the use of FOL is necessary as there is a necessity to learn conceptual knowledge at multiple levels of abstraction – individual object level (ex., the red block), sets of objects (ex., the red blocks) or over all

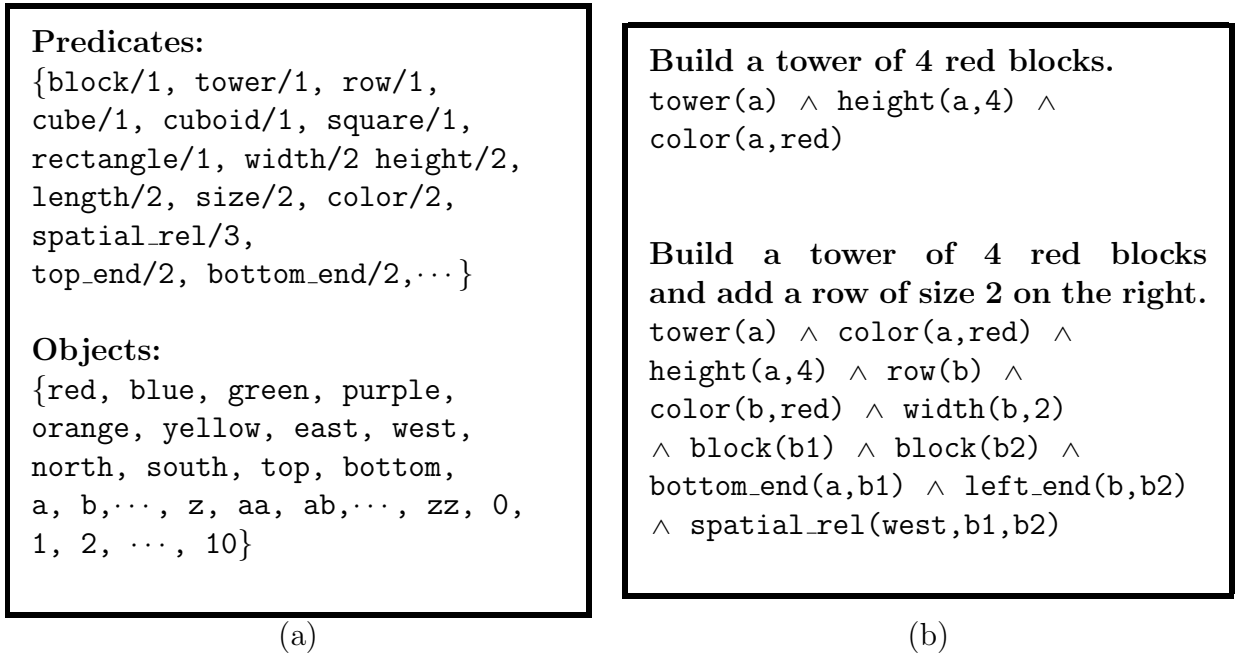


Figure 7.5: FOL representation of collaborative building task. (a) FOL language (Complete list of predicates is deferred to Appendix) (b) Example FOL instructions.

the objects. Learning and reasoning in such a rich space of abstractions is facilitated by the use of ILP and relational planners.

However, while it is possible to represent the complete semantics of the natural language instruction in FOL, it is also clear that for the purposes of this task, a restricted form of FOL would suffice. This restriction is necessary to maintain the tractability of learning and reasoning. The essential knowledge discussed earlier in Section 7.3.2 comprises the predicates of this language. Figure 7.5a presents the first-order language used to describe the collaborative building tasks. Figure 7.5b shows examples of FOL representations of natural language instructions.

To translate the natural language to FOL, we developed two independent NLP parsers: template-based and AMR-based. Our domain-specific **template-based parser** uses a fixed set of templates consisting of slots. These slots are filled by the parser by going through the sentence and looking for matching short phrases. Once the slots are filled, the template is translated into the logic format. It is similar in spirit to *Template Matcher* parser by Jackson et al. (1991), with output in FOL. While our parser is quite fast, it has a few limitations. This parser maintains a *fixed* list of structures known to the builder and their dimensions and thus has quite a limited vocabulary. It does not support all possible phrasing of instruction and requires manual updates of templates to support new sentence formulations. To overcome these limitations, we built an AMR-based parser.

While the template-based parser uses predefined templates for the translation of simpler sentences, the **AMR-based parser** supports free-form sentences of varying complexity. As AMRs can be systematically translated to FOL (Bos, 2016), we use a neural parser to parse natural language text to AMRs. Figure 7.4 ② shows the AMR representation of the sample natural language instruction presented earlier. AMR annotation had not been approached with spatial semantics in mind. Bonn et al. (2020) extends AMR with a spatial addendum, which enables more expressive representation for spatial relationships required in the three-dimensional domain of Minecraft. Minecraft-specific 3D structure building dialogues were

collected between human architect and human builder and annotated with the new inventory of spatial rolesets (Narayan-Chen et al., 2019). A state-of-the-art neural AMR parser, STOG (Zhang et al., 2019), was trained on this Minecraft spatial AMR corpus (Bonn et al., 2020).

### 7.3.6 Planner

This component is responsible for providing the action sequence of placing blocks in the Minecraft environment. Wichlacz et al. (2019) show that a hierarchical planner (see §2.4.2) is better suited than a classical planner for the Minecraft building task. Consequently, a **hierarchical planner**, JSHOP2 (Ilghami, 2006), is employed in this system to generate build plans. JSHOP2 uses a restricted version of HDDL, the same as SHOP2 (Nau et al., 2003), and expects the goal description as a logical combination of predicates defined in the planning domain. For a concise representation, we use the following format to define predicates for all the primitive structures,

```
(structure-name x-location y-location z-location
    [height] [width] [length] color).
```

Following a LISP format, the predicate name is the first element of the tuple which defines the name of the structure. The first three arguments are the X, Y, and Z coordinates of a pivotal block of the structure. The next three arguments represent the three dimensions of the structure. These dimensions, denoted within square brackets are not mandatory for all. Different structures have different dimensions, so the mandatory arguments change accordingly. For example, height is mandatory for a tower. So, the **tower** predicate has 5 arguments. "(tower 0 0 0 4 red)" represents a tower with height 4, color red, and the pivotal block (i.e. lowest block) at the (0,0,0) location in the grid. A complete list of predicates in the planning domain is presented in the Appendix.

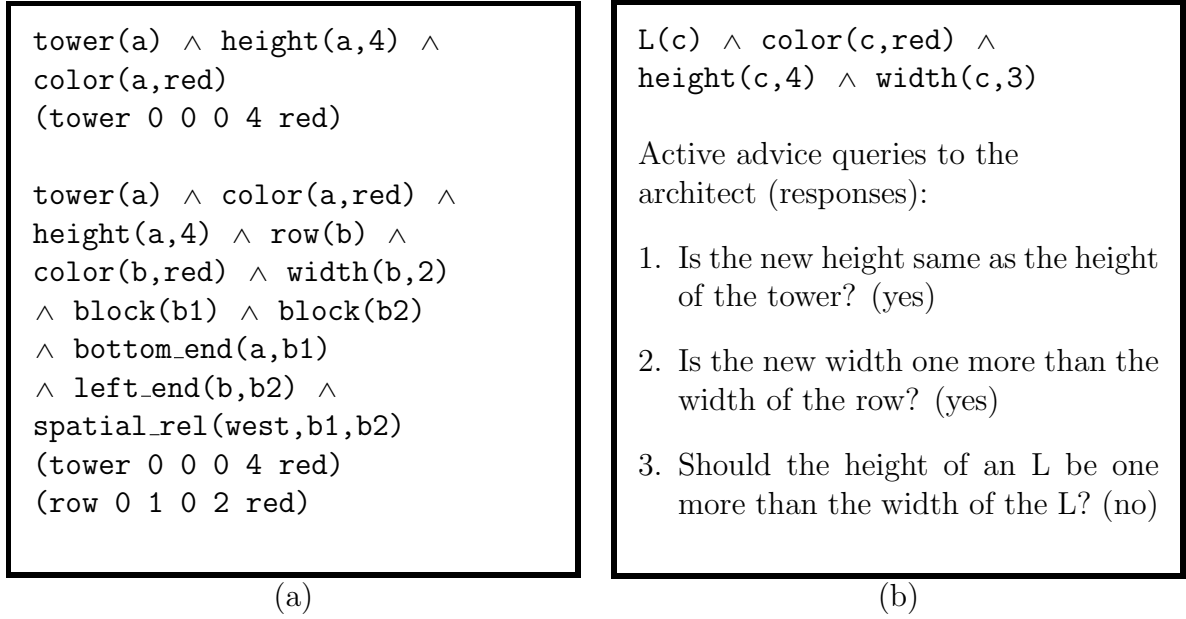


Figure 7.6: (a) Goal description of the FOL instruction. (b) FOL representation of new structure “L” and active advice queries to the architect.

The **logic parser**, translates the FOL instruction to the goal description in the above format. Location coordinates are explicitly constrained within grid boundaries while achieving spatial relations. Figure 7.6 presents example goal descriptions of two FOL instructions presented earlier. When any of the required dimensions are missing, the planner returns an INCOMPLETE GOAL DESCRIPTION error highlighting the missing dimension. In that case, the dialogue manager generates a natural language question for that dimension and completes the goal description from the response.

The **repository** contains all the concept representations known to the builder. Initially, the repository only contains the mapping between the 8 primitive structures and dimensions; identifying the required argument for each primitive structure. Gradually, representations of new structures are also added to the repository (Further details in Section 7.3.7). A complete goal description and an initial state (description of the current build region) are provided to the planner. The hierarchical planner then provides a sequence of block placement or

removal actions to transform the current build region to the goal description. An example plan is presented in Figure 7.4 ④.

### 7.3.7 Concept learner

While there exist various ILP systems that learn concepts (Muggleton and Feng, 1990; Muggleton, 1995; Quinlan, 1990), they all require multiple (positive and negative) examples for each concept. Our collaborative building task, on contrary, requires the agent to learn a new structural concept from just one positive example of the structure. To this effect, we leverage the *Guided One-shot Concept Induction* (GOCI) framework (Das et al., 2020). At its core, GOCI does use an ILP engine to propose candidate hypotheses for a concept. But learning a suitably generalized concept representation from a single example, where infinitely many generalizations are possible, is complex. GOCI uses a powerful inter-representational distance as well as actively solicits advice from humans to prune such a hypothesis space. In our context, queries to obtain advice are posed as simple yes/no questions to the architect. For instance, Figure 7.6b illustrates the FOL representation and the queries by GOCI for an example structure “L” of red color, height of 4, and width of 3, composed of a tower and a row of sizes 4 and 2, respectively. Note that GOCI poses a minimum possible number of queries subject to the complexity of the structure. However, as shown in Das et al. (2020), GOCI poses an average of  $5.5 \pm 3$  queries in Minecraft experiments, which is significantly less than the sample complexity for learning the structural concepts outlined here with vanilla ILP.

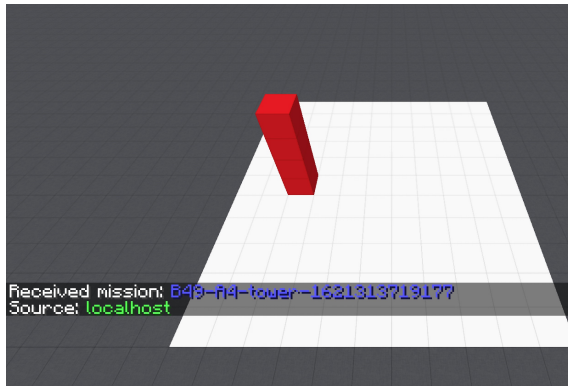
On receiving responses from the architect, GOCI prunes the hypothesis space and finds the most suitable hypothesis which represents the generalized concept of L which is a composition over existing structures. In this current example, the concept L is learned as the

following horn clause,

```
L(X) :- tower(A) ∧ color(A,C) ∧ height(A,H) ∧ row(B) ∧
        color(B,C) ∧ width(B,W) ∧ block(R) ∧ block(S) ∧
        bottom_end(A,R) ∧ left_end(B,S) ∧ spatial_rel(west,R,S) ∧
        color(X,C) ∧ height(X,H) ∧ width(X,V) ∧ one_more(V, W).
```

Uppercase indicates variables and the only constant in this horn clause is **west**. Clausal representations of new structures are stored in the repository. On encountering such non-primitive structures in the instruction, the logical parser would first replace the non-primitive structure with the primitive structures from the clausal theory and then generate a goal description.

**Generalization** is an important factor here. It is not merely a composition of the representations of the sub-concepts, but includes complex decisions about parameter tying, shared logical variables, and partial grounding such that we end up learning a suitable generalization of the given structure and not other relatively similar structures. For instance, the **spatial\_rel()** predicate represents at which relative position the tower and the row connect and in which direction. Hence the clause does not represent other concept classes like “inverted L”. Sammut and Banerji (1986) have, precisely, outlined the nuances of such generalization via logic programs. Another aspect of generalization is learning to grow concept hierarchies in GOCI. For example, once an L structure is learned as a combination of a tower and a row, a U structure can be learned as a combination of an L and a tower. So, the generalization of ‘U’ to a composition of 2 towers and a row is achieved by resolving the hierarchy graph. We demonstrate a concept hierarchy in Section 7.4s. If needed, the architect can also ask the system to forget a learned concept. This can be done by the following command in the chat interface, “Forget <structure-name>”.



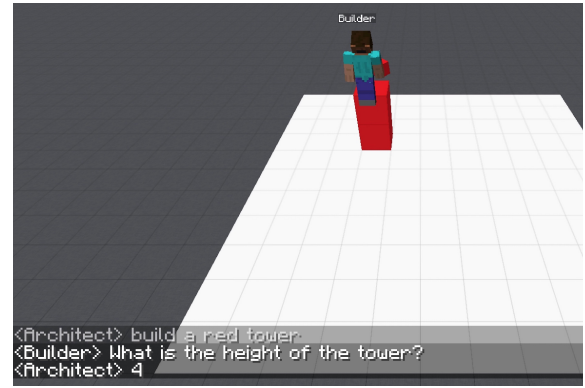
(a)



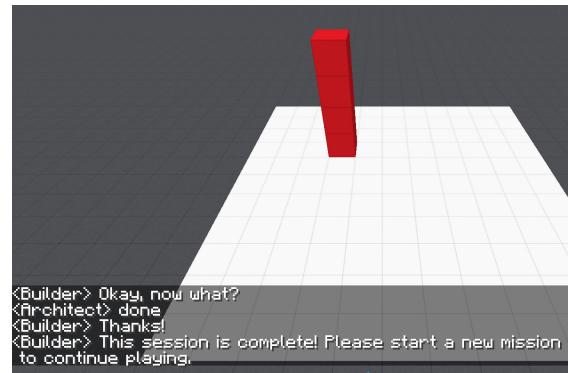
(b)



(c)



(d)

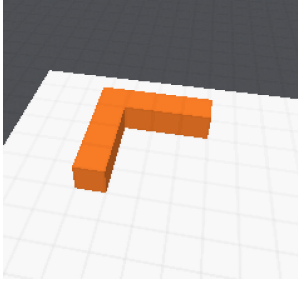


(e)

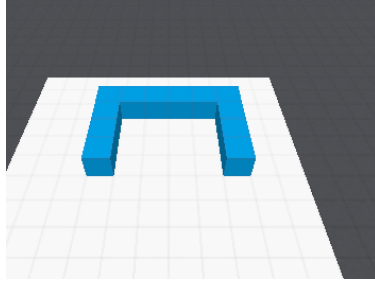
<B> Hello Architect, I'm ready!  
 <A> build a red tower  
 <B> What is the height of the tower?  
 <A> 4  
 <B> Okay, now what?  
 <A> done  
 <B> Thanks!  
 <B> This session is complete! Please start a new mission to continue playing.

(f)

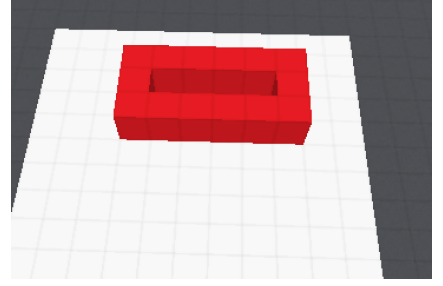
Figure 7.7: Illustration of Lara. (a) A new mission—red tower of height 4. (b) Greeting from the builder and first instruction from the architect to build a red tower. (c) The builder enquires about the height of the tower and the architect provides the height value 4. (d) The builder understands the task, generates a plan, and placed red blocks in the build region. (e) After completing the instruction the builder asks for the next instruction and the architect says done. This ends the mission. (f) The complete chat of this mission is here in text for clarity ‘<B>’ and ‘<A>’ indicates a message from the builder and the architect, respectively.



(a) Gamma structure ( $\Gamma$ )



(b) Cap structure ( $\sqcap$ )



(c) Box structure ( $\square$ )

Build an orange row of size 4.  
Place an orange column of size 3.  
The first block of the column is in front of the left end of the row.  
done

This is an orange Gamma of length 4 and width 4.

(d) Instructions for Gamma

Build a blue gamma of size 4 by 6.  
Place a blue column of size 3 in front of gamma.  
The backend of the column is in front of the left end of the gamma.  
done

This is a Cap of length 4 and width 6 in blue color.

(e) Instructions for Cap

Build a red cap of size 2 by 6.  
Place a red row of size 6 in front of the cap.  
The rightend of the row is in front of the right end of the cap.  
done

This is a box of length 4 and width 6 in blue color.

(f) Instructions for Square

Figure 7.8: Demonstration of the concept hierarchy. (a) Gamma structure ( $\Gamma$ ) made from a row and column. (b) Cap structure ( $\sqcap$ ) made from gamma and a column. (c) Box structure ( $\square$ ) made from a cap and a row. Subfigures (d), (e), and (f) present the natural language instructions.

GOCI adapts Normalized Compression Distance (NCD) (Goldman and Kuter, 2015) to measure the similarity between concept representations. While successful in a few important domains, NCD is limited. Appendix C.4 discusses the limitations of NCD and how they were overcome in Lara.

## 7.4 Demonstration

Figure 7.7 demonstrates our system. We present screenshots of a mission of building a red tower from the beginning till the end. Figure 7.7f reproduces the complete natural language conversation between the builder and the architect.

Further capabilities of Lara are demonstrated in the videos available from the following URL: <https://starling.utdallas.edu/papers/lara/>. Here we illustrate the capability of Lara to learn concept hierarchies using the GOCI framework. Figure 7.8 presents three different concepts, at three different levels of the hierarchy. The first structure Gamma (Figure 7.8a) is built as a composition of a row and a column. The natural language instructions for this structure are shown in Figure 7.8d. The next structure, Cap (Figure 7.8b) is composed of a gamma and a column. This forms the second level of the hierarchy. Further, Figure 7.8c shows the third level of the hierarchy where the concept Box is composed of a cap and a row.

## 7.5 Related work

Blocks world and its variants have been used as toy examples in developing AI and automated planning systems for a long time (Nau et al., 1999). Prominently, Winston (1970) used blocks world for learning structural description from examples. SHRDUL, a dialog system by Winograd (1972), used blocks world for natural language understanding in a 3-dimensional world. While many follow-up works have used blocks world in various human-machine interaction settings, our work simulates the blocks world domain in a popular Minecraft environment.

Human-robot interaction has been addressed in the context of collaborative task achievement by various prior works (Blaylock et al., 2003; Clodic et al., 2008; Fiore et al., 2014; Devin and Alami, 2016; Lemaignan et al., 2017; Krishnaswamy et al., 2020). Of these, we find Lemaignan et al. (2017) and Krishnaswamy et al. (2020) most relevant. Lemaignan et al.

(2017) identifies and characterizes the challenges in building a cognitive robot that shares space and tasks with humans. While a few challenges in their setting are similar to ours (for eg. communication and shared vocabulary), in our work humans and robots do not share the space. That is, only robots are able to modify the environment and humans can only monitor it. Krishnaswamy et al. (2020) presents a situated multimodal interactive agent, Diana. Diana understands vocal instructions, gestures, and facial expressions. Like Lara, Diana is situated in a virtual world and interacts with humans to achieve a task. However, Diana’s tasks are limited and do not require long-term planning.

## 7.6 Conclusion

We have considered the problem of human-machine collaboration in the context of a Minecraft task. Our proposed system Lara, is an integration of several important areas of related research. Specifically, it uses NLP for communication with humans, knowledge representation for representing and reasoning with generalized knowledge, inductive logic programming for efficiently learning higher-order concepts, and hierarchical task planning for effective problem-solving. Lara has the following salient features:

- It allows for *rich natural language instructions* from a human architect by employing an NLP-based parser.
- It represents common knowledge in a rich *FOL based knowledge-base that allows for effective generalization while not sacrificing efficient reasoning*.
- It uses an *ILP-based concept learner that efficiently learns hierarchical, generalized concepts* from a very small number of (potentially single) examples.
- It uses a *hierarchical planner that constructs plans in a stage-wise manner* to effectively exploit the concepts learned in earlier interactions.

- It computes its *uncertainty over its plans/concepts* and queries the human expert for additional knowledge.

In effect, the system knows-what-it-knows and solicits information about what it does not know. This additional information is then used to guide the concept learner and the planner in their tasks.

While successful, the system can potentially be improved in several directions. The richness of natural language interaction can be improved by adding more recent neural-based parsers and generators. Allowing for multiple modalities of communication including gestures is an important future direction. Extending the planner and concept learner to handle complex hybrid data by allowing for them to be differentiable is a necessary step. Finally, large-scale evaluation in more complex domains is an interesting future direction.

## **PART IV**

### **LARGE HYPOTHESIS SPACE**

## CHAPTER 8

### LANGUAGE BIAS IN NEUROSymbOLIC MODELS

Inductive Logic Programming (ILP) (Muggleton, 1992) tasks focus on learning interpretable rules (or logic programs) that can classify a set of positive and negative examples in a data-efficient manner. With the advent of neurosymbolic approaches (d’Avila Garcez et al., 2015), there is a renewed interest in addressing ILP tasks with neural architectures. Various approaches have been proposed that convert logical connectives to neural operations for efficient differentiable learning (Yang et al., 2017; Evans and Grefenstette, 2018; Campero et al., 2018; Dong et al., 2019; Sen et al., 2022; Glanois et al., 2022; Badreddine et al., 2022). These approaches have shown to be effective not only in ILP tasks, but also for visual question answering tasks and reinforcement learning tasks (Dong et al., 2019; Glanois et al., 2022). While successful, we note that they are evaluated only on ILP tasks with homogeneous objects. However, many ILP tasks have heterogeneous objects, that is, domains have objects with different types. In this work, we evaluate the neurosymbolic approaches on ILP tasks with heterogeneous objects, identify the lack of type biases, and devise an approach to incorporate it.

### 8.1 Background

#### 8.1.1 ILP task

A first order language  $\mathcal{L}$  consists of a finite number of extensional<sup>1</sup> predicates ( $P$ ), objects or constants ( $C$ ), types ( $T$ ), and variables ( $V$ ). Every predicate has a fixed arity and each argument of the predicate is associated with a type. All of the objects and the variables are associated with a type and the association is defined as a function  $D : T \mapsto 2^C$ . There is a special type  $t_0$  that encapsulates all the objects,  $D(t_0) = C$ . For every pair of types  $t_i, t_j \in T$ ,

---

<sup>1</sup>As opposed to *intensional* predicates that are derived by a set of clauses.

either  $D(t_i) \subseteq D(t_j)$ , or  $D(t_i) \supseteq D(t_j)$ , or  $D(t_i) \cup D(t_j) = \emptyset$ . An atom is a predicate symbol followed by a parenthesized list of arguments,  $predicate(term1, term2, \dots)$ . The arguments of the predicate can be constants or variables. A task in ILP is then defined over the first order language as a four tuple  $(p_t, \mathcal{B}, \mathcal{P}, \mathcal{N})$ , where  $p_t \in P$  is a target concept or predicate to be learned,  $\mathcal{B}$  is a set of ground atoms representing the domain information (often called as background knowledge),  $\mathcal{P}$  is a set of ground atoms representing positive examples of the target concept, and  $\mathcal{N}$  is a set of ground atoms representing negative examples of the target concept. Given the ILP task, a learning algorithm has to learn a hypothesis—an inductive logic program—to correctly classify the examples.

### 8.1.2 Biases in ILP

As the hypothesis space of the inductive logic program can be too large to handle, language or search biases are used to prune the hypothesis space (or the search space). These biases include various syntactic and semantic restrictions, for example, defining maximum predicate arity, limiting logic programs to definite clauses (or horn clause; clauses with exactly one positive literal), defining the maximum number of literals in the body of a clause, limiting the number of new terms used in the clause, etc. Tausend (1994) studied the use of different biases in ILP and their effects on hypothesis space.

To scale the ILP and relational learning methods for larger domains, with heterogeneous objects, stronger biases are used. Specifically, types and modes are declared over predicate definitions (Muggleton, 1992; Raedt, 2008). *Type declarations bind each argument of a predicate to a fixed type.* For example, `authors(A:person, B:paper)`<sup>2</sup> declares the first argument of the `authors` predicate has to be a `person` and the second argument has to be a `paper`. Conforming to type declaration restricts the hypothesis space. For example,

---

<sup>2</sup>Following the prolog notation, we use lowercase for constant identifiers and uppercase for variable identifiers.

given `authors(A:person, B:paper)` and `venue(X:paper, Y:publication)` a clause like `‘authors(P, Q), venue(P, R)’` is prohibited as a variable `P` cannot simultaneously bind to type `person` and `paper`.

*Mode declarations restrict the terms in a clause.* In ILP and relational learning, a clause is constructed by collecting literals one by one, so literals have an inherent order. The mode declarations indicate the restriction on the terms of the candidate literal (the one being added next) given the terms in the clause so far. Modes are declared by assigning one of the three mode symbols to each term of a predicate. Three mode symbols include: input mode `+`, output mode `-`, and ground mode `#`. The input mode restricts the term in the new literal to an existing variable. The output mode indicates that the term in the new literal can be an existing variable or a new variable. The ground mode `#` indicates that the term in the literal can be a ground object or a constant. As type and mode declarations provide significant bias, which enables scaling ILP and relational learning approaches to large domains, various approaches have been proposed to automatically generate these biases (Cabral et al., 2005; Hayes et al., 2017; Picado et al., 2021).

### 8.1.3 Biases in NeSy

Inductive biases are embedded in the neural network architectures to restrict the space of functions. For example, Convolutional neural networks embody a locality bias, recurrent neural networks assume temporal invariance, and graph neural networks assume permutation invariance (d’Ascoli et al., 2019; Cohen and Shashua, 2017; Battaglia et al., 2018). NeSy architectures (where a neural net is used for differentiable ILP) as well embody various biases by borrowing ideas from ILP literature. For example, differentiable ILP ( $\partial$ ILP) (Evans and Grefenstette, 2018) and Neural Logic Machines (NLMs) (Dong et al., 2019) restricts the maximum predicate arity, Lifted Relational Neural Networks (LRNN) (Sourek et al., 2018) restricts the hypothesis space to definite clauses, Logical Rule Induction (LRI) (Campero

et al., 2018) and Hierarchical Rule Induction (HRI) (Glanois et al., 2022) frameworks restrict the maximum number of literals in the body of a clause, etc. Most importantly, NeSy architectures use different types of templates and meta-rules to define the neural architectures, embodying a language bias in the architecture.

Some NeSy architectures used rules (or inductive logic programs) to define the network structure (Towell and Shavlik, 1994; Sourek et al., 2016; Hu et al., 2016; Kazemi and Poole, 2018), while others relied on templates. Evans and Grefenstette (2018) introduced template constraints to generate multiple rules (or clauses) of length two and learn the weights for each rule in  $\partial$ ILP. Each template constraint is of the form  $\tau_p^i = (v, int)$  where  $p$  is the head predicate,  $i$  counter for the template,  $v$  is the number of existential variables, and  $int$  is a boolean indicator if intensional (or auxiliary) predicates allowed in the body of the clause. Campero et al. (2018) uses proto-rule templates in their Logical Rule Induction (LRI) framework. These proto-rule templates restrict combinations of variable arguments in the clause. Like second-order horn clauses, the proto-rule templates use variables for predicates that are assigned to intensional or extensional predicates while training. For example, a chain proto-rule template  $H(X, Y) \leftarrow B_1(X, Z) \wedge B_2(Z, Y)$  allows clause `grandfather(bill, ann)  $\leftarrow$  father(tom, mary)  $\wedge$  mother(mary, ann)` but not `grandfather(bill, ann)  $\leftarrow$  father(tom, ann)  $\wedge$  mother(mary, ann)`.

While such templates and clauses are strong biases, they need some hand-engineering and some knowledge of the domain as well as ILP for successful declarations. Various approaches have been proposed in the literature to avoid the hand-engineering of rules. Kaur et al. (2019) used lifted relational random walks to generate multiple rules. Kaur et al. (2020) proposed learning tree-structured clauses using a statistical relational learning approach. Glanois et al. (2022) proposed a general set of proto-rule  $\mathcal{R}_0$  (in Equation 8.1) and show that it generates a hypothesis space that is exactly the set of function-free definite horn-clause fragment

composed of clauses with at-most two body atoms involving unary and binary predicates.

$$\mathcal{R}_0 = \left\{ \begin{array}{l} \mathfrak{A} : H(X) \leftarrow \overline{B}_1(X, Y) \wedge \overline{B}_2(Y, X) \\ \mathfrak{B} : H(X, Y) \leftarrow \overline{B}_1(X, Z) \wedge \overline{B}_2(Z, Y) \\ \mathfrak{C} : H(X, Y) \leftarrow \overline{B}_1(X, Y) \wedge \overline{B}_2(Y, X) \end{array} \right\} \quad (8.1)$$

This proto-rule template is similar to LRI. However, unlike LRI where a predicate variable in proto-rule can only correspond to (intensional or extensional) predicates of the same arity, a predicate variable in proto-rule of HRI can correspond to (intensional or extensional) predicates of the same or lower arity. For example, a proto-rule template  $H(X, Y) \leftarrow \overline{B}_1(X, Y) \wedge \overline{B}_2(X, Y)$  allows clause `grandfather(bill, ann) ← male(bill) ∧ grandparent(bill, ann)`.

#### 8.1.4 NeSy predicate invention

While various NeSy approaches have been proposed for predicate invention, we build up on the current state-of-the-art approach, HRI by Glanois et al. (2022). The representation model used by HRI is similar to the one used by Rocktäschel and Riedel (2017) and Campero et al. (2018), however, they significantly improve the training procedure. A predicate  $p \in P$  is represented as a  $d$  dimensional embedding  $\theta_p \in \mathbb{R}^d$ . An atom  $a = p(s, o)$  is represented as a four tuple  $(\theta_p, s, o, v_a)$ , where  $\theta_p$  is the predicate embedding,  $s$  and  $o$  are objects in the atom, and  $v_a \in [0, 1]$  is the valuation that estimates the belief of  $a$  being **True**. The soft unification is computed using cosine renormalized with a softmax transformation. The similarity between two (intensional or extensional) predicates  $p$  and  $B_1$  is computed as,

$$\alpha_{p, B_1} = \frac{\exp(\cos(\theta_p, \theta_{B_1})/\tau)}{\sum_{p'} \exp(\cos(\theta_{p'}, \theta_{B_1})/\tau)}, \quad (8.2)$$

where  $p'$  are all the candidate predicates and  $\tau$  is temperature hyperparameter that controls the renormalization. A hierarchical architecture is defined with one extensional predicate at each layer for every proto-type template, as shown in the Figure 8.1. A proto-rule template (like  $\mathfrak{A}$  from Equation 8.1) is represented as a tuple of embedding  $\mathfrak{R} = (\theta_{\mathfrak{R}}^1, \theta_{\mathfrak{R}}^2, \theta_{\mathfrak{R}}^3) \in \mathbb{R}^d \times$

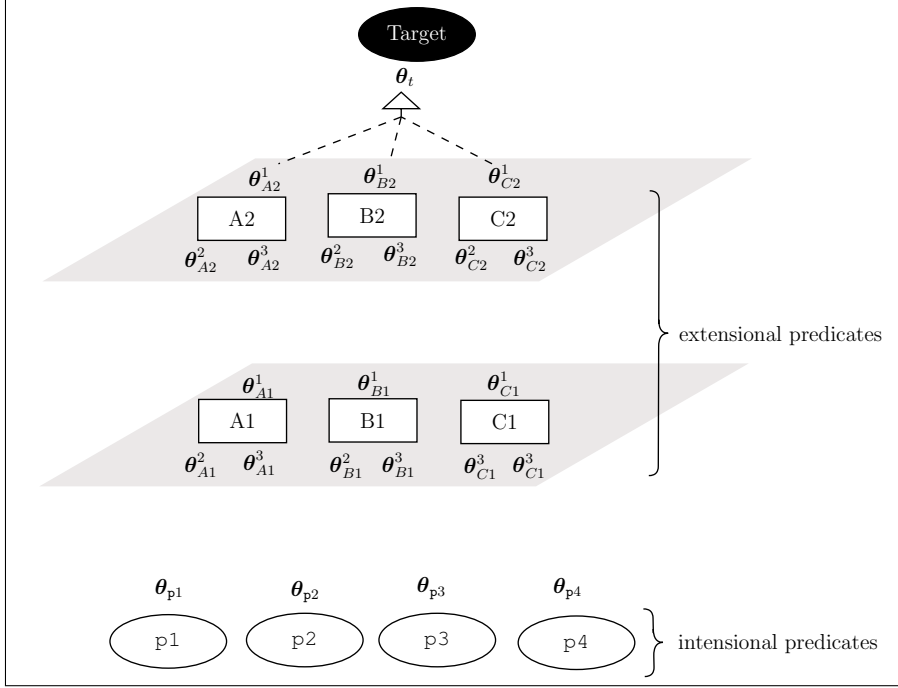


Figure 8.1: HRI architecture.

$\mathbb{R}^d \times \mathbb{R}^d$ , with one predicate embedding for each predicate in the head and the body. Each body predicate in the proto-rule is connected with the head of the extensional predicates appearing at the same or lower level as well as all the intensional predicates.

A rule can be evaluated for two facts  $\mathbf{f1} = (\theta_{p1}, \mathbf{s}_{f1}, \mathbf{o}_{f1}, v_{f1})$  and  $\mathbf{f2} = (\theta_{p2}, \mathbf{s}_{f2}, \mathbf{o}_{f2}, v_{f2})$  if they satisfy the variable arguments combination specified in the rule. When it does, the value of this evaluation is defined as follows,

$$v(\mathfrak{R}, \mathbf{f1}, \mathbf{f2}) = \alpha_{p1B1} \cdot \alpha_{p2B2} \cdot \text{AND}(v_{f1}, v_{f2}) \quad (8.3)$$

The valuation of an extensional atom is, however, computed by pooling across different combinations of the facts and merging with the old valuation. Hence,

$$v^{out}(\mathfrak{R}) = \text{MERGE} \left( v_{old}^{out}, \text{POOL}_{\mathbf{f1}, \mathbf{f2}} (\alpha_{p1B1} \cdot \alpha_{p2B2} \cdot \text{AND}(v_{f1}, v_{f2})) \right) \quad (8.4)$$

The POOL operator involves maximizing over the existential variable (if any) and summing over the different facts. The MERGE operator is implemented as max. The valuation of the target predicate is computed similarly, by merging and pooling the extensional predicate at the last layer ( $P_l$ ),

$$v^t = \text{MERGE} \left( v_{old}^t, \text{POOL}_{P \in P_l} (\alpha_{P_{\mathbf{p}_t}} \cdot v_P) \right). \quad (8.5)$$

The model is trained with binary cross entropy loss with a regularization that encourages interpretability. We refer the readers to Glanois et al. (2022) for further details.

## 8.2 Introducing Typed Bias

We first describe two different ways of introducing type bias in the model, and then evaluate the approaches.

### 8.2.1 Approach I

The first approach is inspired by the invalid action masking (Huang and Ontañón, 2022) technique used in Reinforcement Learning. In complex RL with large discrete actions, many actions are invalid in specific states. In such cases, policy gradient algorithms use invalid action masking technique to mask out the invalid action and only consider the scores of the valid action for sampling. This technique has been widely adopted in many RL frameworks. Huang and Ontañón (2022) shows that the mask can be considered as a state-dependent differentiable function and the masking of the invalid actions is equivalent to making the gradients of the invalid actions zero. Inspired by this, we consider the masking of target atoms with invalid constant types. For example, if a binary target predicate  $\mathbf{p}_t$  has argument types  $t_i$  and  $t_j$ , then the valuation  $v^t$  is masked out (set to zero) for all combinations of constants that do not conform to the specified types,  $\{(s, o) | s \notin D(t_i), o \notin D(t_j)\}$ . We call this approach Masked-HRI.

### 8.2.2 Approach II: HTRI

The second approach uses type embeddings for hierarchical typed rule induction (HTRI). Here, every type in the first-order language is represented as an embedding  $\delta \in \mathbb{R}^{|T|}$ . The similarity between two predicates is indicated as a similarity between the predicate embedding concatenated with type embedding, as follows,

$$\alpha_{\mathbf{p}, B_1} = \frac{\exp(\cos(\boldsymbol{\theta}_{\mathbf{p}} \parallel \boldsymbol{\delta}_{\mathbf{p}}^s \parallel \boldsymbol{\delta}_{\mathbf{p}}^o, \boldsymbol{\theta}_{B_1} \parallel \boldsymbol{\delta}_{B_1}^s \parallel \boldsymbol{\delta}_{B_1}^o) / \tau)}{\sum_{\mathbf{p}'} \exp(\cos(\boldsymbol{\theta}_{\mathbf{p}'} \parallel \boldsymbol{\delta}_{\mathbf{p}'}^s \parallel \boldsymbol{\delta}_{\mathbf{p}'}^o, \boldsymbol{\theta}_{B_1} \parallel \boldsymbol{\delta}_{B_1}^s \parallel \boldsymbol{\delta}_{B_1}^o) / \tau)}, \quad (8.6)$$

Proto-rules are redefined as follows with types as shown in Equation 8.7. Each rule is represented as  $\mathfrak{R} = (\theta_{\mathfrak{R}}^1, \theta_{\mathfrak{R}}^2, \theta_{\mathfrak{R}}^3, \delta_{\mathfrak{R}}^1, \delta_{\mathfrak{R}}^2, \delta_{\mathfrak{R}}^3) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{|T|} \times \mathbb{R}^{|T|} \times \mathbb{R}^{|T|}$ . Hence, each rule-learning involves learning body predicates as well as the associated types.

$$\mathcal{R}_0 = \left\{ \begin{array}{l} \mathfrak{A} : \text{H}(X) : (T1) \leftarrow \overline{\text{B}}_1(X, Y) : (T1, T2) \wedge \overline{\text{B}}_2(Y, X) : (T2, T1) \\ \mathfrak{B} : \text{H}(X, Y) : (T1, T2) \leftarrow \overline{\text{B}}_1(X, Z) : (T1, T3) \wedge \overline{\text{B}}_2(Z, Y) : (T3, T2) \\ \mathfrak{C} : \text{H}(X, Y) : (T1, T2) \leftarrow \overline{\text{B}}_1(X, Y) : (T1, T2) \wedge \overline{\text{B}}_2(Y, X) : (T2, T1), \end{array} \right\} \quad (8.7)$$

We use the following binary cross entropy loss as the training objective with a regularizer that encourages unification scores to be closer to 0 or 1.

$$\sum_{s,o} -G_{\mathbf{p}_{\mathbf{t}}(s,o)} \log(v_{\mathbf{p}_{\mathbf{t}}(s,o)}) - (1 - G_{\mathbf{p}_{\mathbf{t}}(s,o)}) \log(1 - v_{\mathbf{p}_{\mathbf{t}}(s,o)}) + \lambda \sum_{P,P'} \alpha_{P,P'} (1 - \alpha_{P,P'}) \quad (8.8)$$

## 8.3 Experiments

In our experiments, we aim to evaluate the following questions:

**Q1.** Are approaches that use type-information effective?

**Q2.** Can we remove the mask during the evaluation?

We evaluate the framework for five tasks of learning operator preconditions. We used the initial state of 100 planning problems for training. Our implementation uses pyper-planner(Alkhazraji et al., 2020) to parse the planning problem and generate the dataset.

We train the model on small problem sets and evaluate it on larger problems. The target predicate, problem sizes, and background predicate details are provided in Table 8.1.

Figure 8.2 compares the hierarchical rule induction (HRI) with the two proposed approaches, Masked-HRI and HTRI on the five tasks. We compare approaches for Recall and Precision, as the task is significantly imbalanced. Our initial experiments reveal that while the HTRI approach performs similarly to HRI in most tasks, the Masked-HRI approach is significantly better. To answer Q1, while using the type information is effective in Masked-HRI, it was not significantly effective in the HTRI approach. Hence, further investigation into different approaches to using the type-information is required.

Further, to investigate the effectiveness of the masking approach, we remove the masking during evaluation. That is, we only mask the invalid valuations during training but do not mask it during testing. Figure 8.3 compares Masked-HRI (that uses mask for training and evaluation) with Masked(T)-HRI (that uses mask only during training). As evident in the

Table 8.1: Summary of the tasks

	<b>Bg. Predicates</b>	<b>Objects</b>	<b>Task</b>	<b>Precondition</b>
Logistics	{in-city, at, in, truck, airplane, package, vehicle, airport, location, city, place, physobj}	Train: 41 Eval: 56	drive-truck(X, Y)	{truck(X), location(Y), at(X, Z), in-city(Z, C), in-city(Y, C)}
			fly-airplane(X, Y)	{airplane(X), airport(Y)}
			load-truck(X, Y)	{package(X), truck(Y) (at X L), (at Y L)}
Satellite	{ on_board, supports, pointing, power_avail, power_on, calibrated, calibration_target, satellite, direction, instrument, mode, have_image }	Train: 19 Eval: 56	turn_to(X, Y)	{satellite(X), direction(Y)}
			switch_on(X, Y)	{instrument(X), satellite(Y), power_avail(Y), on_board(X, Y)}

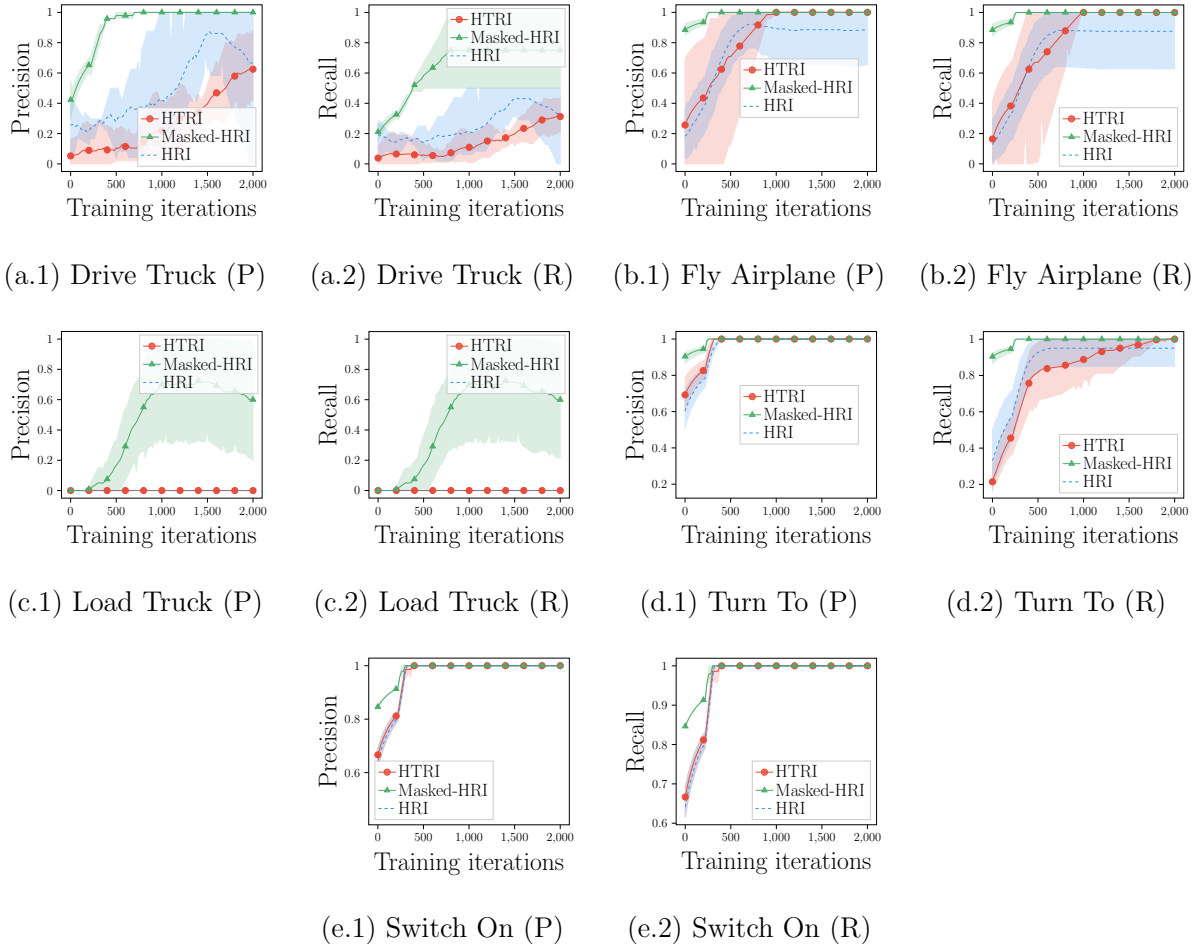
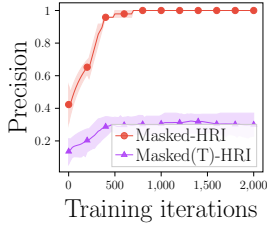


Figure 8.2: Learning curves comparing precision and recall, of HRI with HTRI and Masked-HRI, for learning operator preconditions.

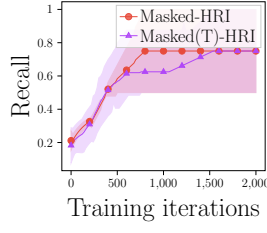
results, using the mask for training alone results in a significant reduction of precision. This answers Q2 in that masking should not be removed during evaluation.

## 8.4 Discussion

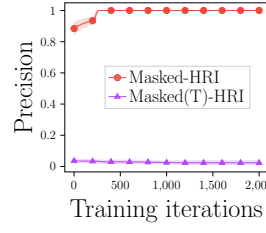
Our initial experiments suggest that using type information to mask the invalid predicates is more effective than using type as embedding. However, as evident from results in Figure 8.3, the rules induced by masking the invalid type predicates are not inherently better. That is the mask is still required for evaluation. So we aim to further modify the learning objective and



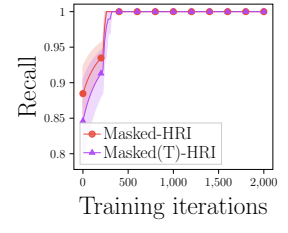
(a.1) Drive Truck (P)



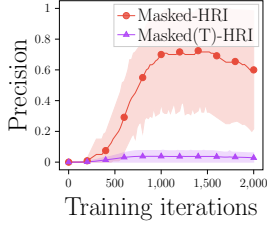
(a.2) Drive Truck (R)



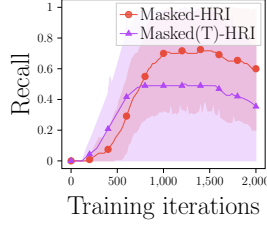
(b.1) Fly Airplane (P)



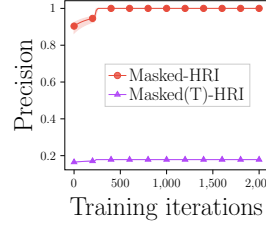
(b.2) Fly Airplane (R)



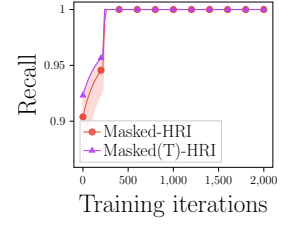
(c.1) Load Truck (P)



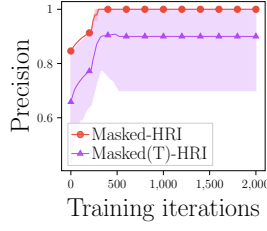
(c.2) Load Truck (R)



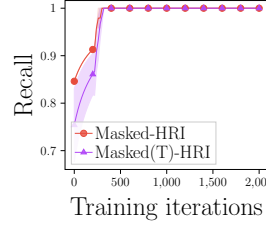
(d.1) Turn To (P)



(d.2) Turn To (R)



(e.1) Switch On (P)



(e.2) Switch On (R)

Figure 8.3: Learning curves comparing precision and recall, of Masked-HRI and Masked(T)-HRI, for learning operator preconditions.

explore different similarity measures for unification to improve the rule-induction. Further, we plan to add qualitative analysis of the induced rules.

## **PART V**

### **OTHER EXPLORATIONS**

## CHAPTER 9

### EXTRACTING QUALITATIVE KNOWLEDGE

Qualitative influence statements are often provided a priori to guide learning (see §2.3). In this chapter, we answer a challenging reverse task of automatically extracting qualitative influence statements from data. We propose an approach, called QuAKE (Karanam et al., 2021), to learn a probabilistic model from the data and then compute the qualitative influences. We apply our qualitative knowledge extraction method to a clinical study for early prediction of adverse pregnancy outcomes—nuMoM2b. Our empirical results demonstrate that the extracted rules are both interpretable and valid.

#### 9.1 Introduction

The nuMoM2b (Nulliparous Pregnancy Outcomes Study: Monitoring Mothers-to-Be) study (Haas et al., 2015) aims to identify early warning signs of adverse pregnancy outcomes, design interventions, and assist with decision-making. Since 2010, eight research sites in the United States followed up with women throughout their pregnancies—collecting routine clinical information, exercise data, and food they ate. Using this data, we consider learning to explain the relationship between gestational diabetes mellitus (GDM) and some common risk factors.

A common way to employ knowledge in machine learning and AI is via the use of qualitative relationships that express how changes in a feature or a risk factor affect the target. These rules were mainly used as “inductive bias” apriori to learning since they are both intuitive and natural in many domains. We address the challenging “reverse task” of extracting these rules from the data. To this effect, in the context of nuMoM2b, we propose a two-step process. First, we learn a joint probability distribution over all the variables including the target (GDM). In the second step, the constraints are extracted by reasoning over this joint

probability distribution. We demonstrate in our experiments that such an approach yields rules that are both intuitive and valid (as validated by our clinical expert Dr. David Haas).

The rest of the chapter is organized as follows. In Section 9.2 we outline our approach called QuaKE. In Section 9.3 we describe the nuMoM2b data and present the extracted rules. Finally, Section 9.4 concludes the chapter with a summary.

## 9.2 Extracting qualitative influences

**Given:** A data set  $\mathcal{D}$  consisting of examples in the form of risk factors  $\mathbf{X}$  and binary target  $Y$  (in this case GDM).

**To Do:** Learn a set of QIs that explain the effect of  $\mathbf{X}$  on  $Y$ .

We use  $X_a$  to denote the  $a^{th}$  variable in the feature set  $\mathbf{X}$ .  $x_a^i$  denotes a particular value of variable  $X_a$  and  $|X_a|$  denotes the number of discrete values  $X_a$  takes. We assume that the joint distribution ( $P$ ) over the set of random variables  $\mathbf{X}$  is known (we learn this joint distribution in our empirical evaluation using a causal learning algorithm). For brevity, we restrict the description of our method to extracting positive MIs and SIs,  $<^{M+}$  and  $<^{S+}$ . The *degree of monotonic influence*,  $\delta_a$ , of  $X_a \in \mathbf{X}$  on  $Y$  is defined as

$$\delta_a = I_{(C_a > 0)} \cdot \sum_j \sum_{j' > j} \sum_k \frac{P(Y \leq k | X_a = x_a^j) - P(Y \leq k | X_a = x_a^{j'})}{|X_a|} \quad (9.1)$$

where,

$$C_a = \prod_j \prod_{j' > j} \prod_k \max(P(Y \leq k | X_a = x_a^j) - P(Y \leq k | X_a = x_a^{j'}) + \epsilon_m, 0) \quad (9.2)$$

For monotonicity to hold, we require  $P(Y \leq k | X_a = x_a^j) + \epsilon_m \geq P(Y \leq k | X_a = x_a^{j'})$  for all pairs of configurations of  $X_a$ ,  $(j, j')$  with  $j' > j$  at any given threshold value  $k$ . Here the monotonic slack  $\epsilon_m$  allows violating a constraint within a chosen margin. The degree of MI,  $\delta_a$ , in Equation 9.1 measures the cumulative difference in the probability that the target variable  $Y$  is less than a threshold  $k$  given  $X_a$  at two different values  $x_a^j$  and  $x_a^{j'}$ .

We extend the concept of degree of MI to SI by conditioning on a pair of variables instead of a single variable. First, consider the difference in the effect of changing  $X_a$  from  $x_a^i$  to  $x_a^{i'}$  on  $Y$  under the context of two different values of  $X_b$  ( $x_b^j$  and  $x_b^{j'}$ ). We define this as

$$\begin{aligned} \phi_{a,b}^{i,i',j,j'} = & \sum_k P(Y \leq k | X_a = x_a^i, X_b = x_b^j) - P(Y \leq k | X_a = x_a^{i'}, X_b = x_b^j) - \\ & P(Y \leq k | X_a = x_a^i, X_b = x_b^{j'}) + P(Y \leq k | X_a = x_a^{i'}, X_b = x_b^{j'}) \end{aligned} \quad (9.3)$$

For synergy to hold, we require  $\phi_{a,b}^{i,i',j,j'} + \epsilon_s$  to be non-negative for all  $i' > i$  and  $j' > j$ . Where  $\epsilon_s$  is the synergistic slack. We define the *degree of synergistic influence*,  $\delta_{a,b}$ , of variables  $X_a \in \mathbf{X}$  and  $X_b \in \mathbf{X}$  on  $Y \in \mathbf{X}$  as the cumulative difference in degrees of context-specific influence of  $X_a$  on  $Y$  in the context of  $X_b$ . It is given by

$$\delta_{a,b} = I_{(C_{a,b} > 0)} \cdot \sum_i \sum_{i' > i} \sum_j \sum_{j' > j} \frac{\phi_{a,b}^{i,i',j,j'}}{|X_a| \cdot |X_b|} \quad (9.4)$$

where,

$$C_{a,b} = \prod_i \prod_{i' > i} \prod_j \prod_{j' > j} \max(\phi_{a,b}^{i,i',j,j'} + \epsilon_s, 0) \quad (9.5)$$

We employ both definitions to learn QIs in Algorithm 4, Qualitative Knowledge Extraction (QuaKE). The algorithm assumes the existence of a joint distribution (Pearl, 1988) over ordinal features, which we learn using a causal probabilistic learning algorithm (PC) (Spirtes and Glymour, 1991; Colombo and Maathuis, 2014). We chose the PC algorithm to verify our hypothesis that the use of a causal model will yield causally interpretable qualitative relationships. We calculate the degree of MI of every variable  $X_a \in \mathbf{X}$  on  $Y$  and SI of every pair of variables  $X_a, X_b \in \mathbf{X}$  on  $Y$ . The MI rules  $X_{a <}^{M+} Y$  are extracted if their corresponding degree of MI  $\delta_a$  is above a pre-defined threshold  $T_m$ . Similarly, the synergistic rules  $X_a, X_{b <}^{S+} Y$  are extracted if their corresponding degree of SI  $\delta_{a,b}$  are above a pre-defined threshold  $T_s$ .

---

**Algorithm 4** QuaKE

---

INPUT: Probabilistic model  $P$ , Target variable  $Y$ , Features  $\mathbf{X}$ , monotonic slack  $\epsilon_m$ , synergistic slack  $\epsilon_s$ , monotonic threshold  $T_m$ , synergistic threshold  $T_s$

OUTPUT: Rules  $\mathbf{R}$

```
1: function QuaKE( $P, Y, \mathbf{X}, \epsilon_m, \epsilon_s, T_m, T_s$ )
2:    $\mathbf{R} \leftarrow \emptyset$  ▷ Initialize rules
3:   for  $a \leftarrow 0$  to  $(|\mathbf{X}| - 1)$  do
4:     compute  $\delta_a$  using Eq. 9.1 ▷ compute the degree of monotonic influence
5:     if  $\delta_a \geq T_m$  then
6:        $\mathbf{R} \leftarrow (X_{a<}^{M+Y}) \cup \mathbf{R}$  ▷ If higher than threshold, add it to rules
7:     end if
8:     for  $b \leftarrow a + 1$  to  $(|\mathbf{X}| - 1)$  do
9:       compute  $\delta_{a,b}$  using Eq. 9.4 ▷ degree of synergistic influence
10:      if  $\delta_{a,b} \geq T_s$  then
11:         $\mathbf{R} \leftarrow (X_a, X_{b<}^{S+Y}) \cup \mathbf{R}$  ▷ If higher than threshold, add it to rules
12:      end if
13:    end for
14:  end for
   (similarly for decreasing cases)
15:  return  $\mathbf{R}$ 
16: end function
```

---

### 9.3 Evaluation on nuMoM2b study

In our evaluations we aim to answer the following two questions:

**Q1.** Does QuaKE extract high-quality rules that align with background knowledge in this domain?

**Q2.** Does QuaKE help uncover QI statements in cases where prior knowledge is uncertain?

#### 9.3.1 The nuMoM2b study

The nuMoM2b study tracked pregnancies of 10,037 women near 8 sites in the United States. We excluded 817 cases where women were already diagnosed with diabetes and studied the remaining 9,220 women. The database contained more than 7,000 variables across the

Table 9.1: Summary of the 8 variables in nuMoM2b dataset.

Attribute	Type	Categories
GDM Diagnosed	Boolean	True or False
Gravidity	Ordinal	1, 2, 3+
Ever used tobacco	Boolean	True or False
Smoked in the last three months	Boolean	True or False
Highest education level completed	Ordinal	Six levels from High School to post-graduate
Race Category	Category	Non-Hispanic White, Non-Hispanic Black, Hispanic, American Indian, Asian, Native Hawaiian, Other, Multiracial.
Age	Ordinal	$< 21$ , $21-25$ , $25-29$ , $29-32$ , $\geq 32$
BMI	Ordinal	low, medium, high

study participants. It’s unlikely that specific knowledge exists for every factor’s influence on Gestational Diabetes Mellitus (GDM), so we first performed feature selection. We took the intersection of features selected for discriminatively predicting  $P(Y = GDM|X)$  with recursive feature elimination and those found by Lasso—then added *Gravidity* and *Education* to this set. *Gravidity* was an influential variable in a previous study that mined electronic health records for GDM risk factors,<sup>1</sup> *Education* can be a weak indicator of socioeconomic status and we believed there could be background knowledge of how it influenced other factors. This resulted in the set of 8 features in Table 9.1.

We interpreted all variables as ordinal since monotonicity and synergy deal with increasing values (e.g. “BMI increasing implies GDM increasing”). For ordinal variables, this was implicit. For the boolean variables, increasing meant  $False \rightarrow True$ . We assume *Race* is categorical, but use an ordering based on previous studies (Hedderson et al., 2010) on the effect of *Race* on *GDM*. This ordering was: Non-Hispanic White, Non-Hispanic Black, Hispanic, American Indian, Asian, Native Hawaiian, Other, and Multiracial.

---

<sup>1</sup>Gravidity did not appear to be an informative factor during our feature selection. The study mentioned focused on GDM risk factors for women with parity  $\geq 0$ , whereas the nuMoM2b population was nulliparous (parity = 0). Combining these two pieces of information may suggest that gravidity is only informative in a parity  $> 0$  population, but we cannot verify this with the data we have available.

### 9.3.2 Setup and baselines

Algorithm 4 has four hyperparameters. We chose the following settings:  $\epsilon_m = 0.003$ ,  $\epsilon_s = 0.02$ ,  $T_M = 0.005$ ,  $T_S = 0.001$ . We compare learned rules with those from our clinical expert, *Dr. Haas*. W.r.t GDM, these could either be increasing, decreasing, no effect, or unknown. Since Algorithm 4 assumes a complete joint distribution  $P$  is available, we consider two factorizations of  $P$ . The first learns a causal model (Colombo and Maathuis, 2014) and the other (baseline) estimates the probabilities directly from data. Alternative baselines might have included rules extracted from decision trees, rule mining, or Bayesian rule learning—but each induces conjunctive rules of the form  $(x_1 \wedge x_2 \wedge \dots \wedge x_n) \implies y$ , making their exact connection to the QI statements tenuous.

### 9.3.3 Results

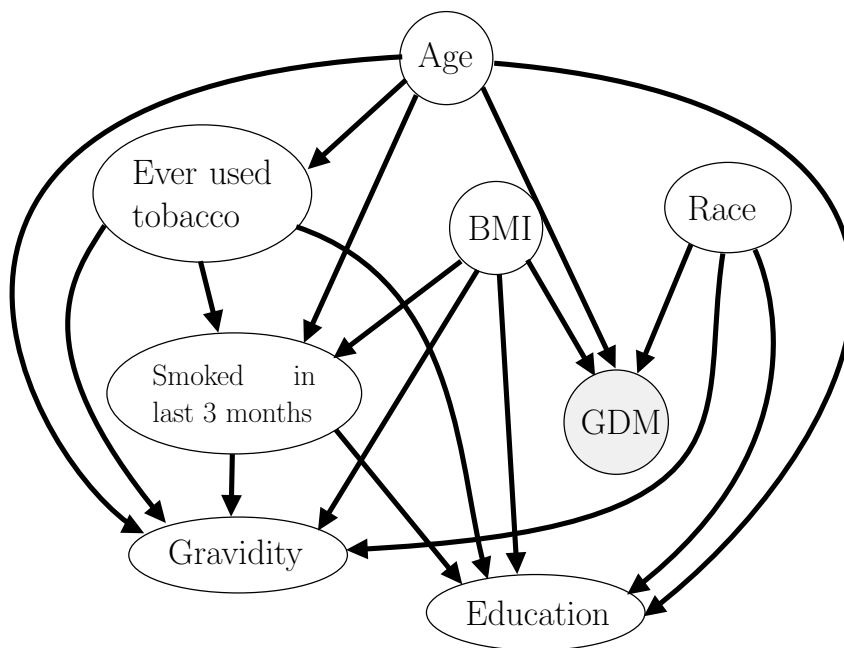


Figure 9.1: Causal network obtained by the stable PC algorithm for the nuMoM2b dataset.

Figure 9.1 shows the causal network obtained by the stable PC algorithm. We use this causal network to learn the joint probability distribution. The rules learned by the QuaKE framework are presented in Table 9.2. The “Prior” knowledge refers to the rules provided by our expert. We compare these to the rules extracted by QuaKE and baseline (Data Alone). QuaKE’s precision compared to expert advice is  $0.923 \pm 0$ ; whereas the precision of our unstructured baseline is  $0.636 \pm 0$ . The precision of each method was consistent across five stratified cross-validation folds. This affirms Q1—QuaKE can extract high-quality rules aligning with prior knowledge.

Since we have formalized the degree of the QIs in Equations 9.1 and 9.4, we can analyze rules that were highly uncertain according to the prior knowledge. Two of the synergistic relations involving smoking and education had an unknown effect in relation to GDM. *Education*, *Smoked\_in\_3\_months* $_{<}^{S+}$ *GDM* was a high-confidence rule extracted by QuaKE and the baseline. We speculate that this could be either due to the high correlation between *Education* and *Age*, or related to an unobserved relationship between education and socioeconomic status. Note that both these results are especially interesting since we found only a weak monotonic relationship between smoking and GDM more generally. We use this to answer Q2—our approach can identify potentially interesting cases where prior knowledge is uncertain.

## 9.4 Summary

We considered the problem of learning interpretable and explainable qualitative rules for modeling GDM. To this effect, we learned a causal (probabilistic) model and recovered the knowledge by applying the rules. Our results indicate that most of our rules are in line with the prior knowledge of our expert and some interesting influence relationships appear that are worth investigating. Incorporating richer domain knowledge, automatically refining

the rules, identifying broader relationships, and scaling to larger feature sets are interesting future research directions.

Table 9.2: Comparison of QI from prior knowledge (PK), QuaKE, and Data Alone.  $\checkmark/\times$  represents that this relationship does/not exist respectively while  $?$  represents unknown influence. The three groups of rows show (1) MI, (2) SI, and (3) sub-SI. Colors highlight rules recovered by QuaKE and show (a.) coherent with the PK and baseline in white (b.) contradicting the baseline in green (c.) coherent with baseline but contradict the PK in blue.

Rule	Prior Knowledge	QuaKE	Data Alone
$\text{BMI}_{<}^{M+}\text{GDM}$	$\checkmark$	$\checkmark$	$\checkmark$
$\text{Age}_{<}^{M+}\text{GDM}$	$\checkmark$	$\checkmark$	$\checkmark$
$\text{Race}_{<}^{M+}\text{GDM}$	$\checkmark$	$\checkmark$	$\times$
$\text{Education}_{<}^{M+}\text{GDM}$	$\checkmark$	$\checkmark$	$\times$
$\text{Gravidity}_{<}^{M+}\text{GDM}$	$\checkmark$	$\checkmark$	$\times$
$\text{Smoked in 3 months}_{<}^{M+}\text{GDM}$	$\checkmark$	$\times$	$\times$
$\text{Used Tobacco}_{<}^{M+}\text{GDM}$	$\checkmark$	$\times$	$\times$
$\text{Age, BMI}_{<}^{S+}\text{GDM}$	$\checkmark$	$\checkmark$	$\checkmark$
$\text{Age, Smoked in 3 months}_{<}^{S+}\text{GDM}$	$\checkmark$	$\checkmark$	$\checkmark$
$\text{BMI, Used Tobacco}_{<}^{S+}\text{GDM}$	$\checkmark$	$\checkmark$	$\checkmark$
$\text{Education, Smoked in 3 months}_{<}^{S+}\text{GDM}$	$?$	$\checkmark$	$\checkmark$
$\text{BMI, Gravidity}_{<}^{S+}\text{GDM}$	$\checkmark$	$\checkmark$	$\times$
$\text{BMI, Smoked in 3 months}_{<}^{S+}\text{GDM}$	$\checkmark$	$\times$	$\checkmark$
$\text{Age, Used Tobacco}_{<}^{S+}\text{GDM}$	$\checkmark$	$\times$	$\times$
$\text{BMI, Education}_{<}^{S+}\text{GDM}$	$\times$	$\checkmark$	$\checkmark$
$\text{Education, Used Tobacco}_{<}^{S+}\text{GDM}$	$?$	$\times$	$\times$
$\text{Age, Education}_{<}^{S-}\text{GDM}$	$\checkmark$	$\checkmark$	$\checkmark$
$\text{BMI, Smoked in 3 months}_{<}^{S-}\text{GDM}$	$\times$	$\checkmark$	$\times$
$\text{Age, Used Tobacco}_{<}^{S-}\text{GDM}$	$\times$	$\times$	$\checkmark$
$\text{BMI, Gravidity}_{<}^{S-}\text{GDM}$	$\times$	$\times$	$\checkmark$
$\text{Gravidity, Used Tobacco}_{<}^{S-}\text{GDM}$	$\times$	$\times$	$\checkmark$
$\text{Education, Used Tobacco}_{<}^{S-}\text{GDM}$	$?$	$\times$	$\checkmark$
$\text{Age, Gravidity}_{<}^{S-}\text{GDM}$	$\checkmark$	$\times$	$\times$

## CHAPTER 10

### DATA SUBSET SELECTION FOR DOMAIN ADAPTATION

Real-world machine-learning applications require robust models that generalize well to distribution shift settings, which is typical in real-world situations. Domain adaptation techniques aim to address this issue of distribution shift by minimizing the disparities between domains to ensure that the model trained on the source domain performs well on the target domain. Nevertheless, the existing domain adaptation methods are computationally very expensive. In this work, we aim to improve the efficiency of existing supervised domain adaptation (SDA) methods by using a subset of source data that is similar to target data for faster model training. Specifically, we propose ORIENT, a subset selection framework that uses the submodular mutual information (SMI) functions to select a source data subset similar to the target data for faster training. Additionally, we demonstrate how existing robust subset selection strategies, such as GLISTER, GRADMATCH, and CRAIG, when used with a held-out query set, fit within our proposed framework and demonstrate the connections with them. Finally, we empirically demonstrate that SDA approaches like  $d$ -SNE, CCSA, and standard Cross-entropy training, when employed together with ORIENT, achieve a) faster training and b) better performance on the target data.

#### 10.1 Introduction

The recent success of deep learning frameworks in applications such as image classification (Ciresan et al., 2012), speech recognition (Hershey et al., 2010), and object detection (Geirhos et al., 2018) stems primarily from the availability of large amounts of labeled data. However, due to enormous labeling costs and the need for specialists in specific domains such as medical imaging, it is not always possible to obtain vast amounts of labeled data in all situations. On the contrary, training deep models on limited amounts of data

can lead to poor performance due to overfitting (Arpit et al., 2017). Consequently, where obtaining large amounts of labeled data is difficult for the target domain, closely related domain (source domain) is used to train the model. However, this may result in a deep model with suboptimal performance on the target domain as a result of the distribution shift (Shimodaira, 2000; Ben-David et al., 2010; Ben-David et al., 2006; Torralba and Efros, 2011), i.e., change of data distribution from source domains to target domains. A change in the data distribution often renders the features learned by the model on the source domain irrelevant in the target domain.

To address the problem of distribution shift, many domain adaptation (Wang and Deng, 2018; Patel et al., 2015) and domain generalization (Muandet et al., 2013) techniques have been proposed in recent years. Domain adaptation methods assume that some target domain information is available (usually target data), whereas domain generalization methods do not. Domain adaptation (DA) methods can be categorized into unsupervised (Gong et al., 2012; Ganin and Lempitsky, 2015; Liu and Tuzel, 2016; Tzeng et al., 2017) (using unlabeled target data), semi-supervised (Guo and Xiao, 2012; Yao et al., 2015; Saito et al., 2019; Singh, 2021) (using labeled and unlabeled target data), and supervised (Motiian et al., 2017; Xu et al., 2019; Morsing et al., 2020; Hedegaard et al., 2021) (using labeled target data) methods. UDA methods that do not require any labeled target data assume access to large volumes of unlabeled target data. When providing the same amount of target data, SDA methods are more effective than UDA methods (Motiian et al., 2017). As it is not difficult to obtain a few samples (as less as 2 examples per class) of labeled target data in practice, SDA methods are an attractive approach in scenarios with limited target data. Hence, in this work, we are explicitly focused on the SDA setting.

Recently, various effective SDA methods are proposed (Motiian et al., 2017; Xu et al., 2019; Morsing et al., 2020). However, they are usually compute-intensive. For example, using one of the state-of-the-art SDA methods, *d*-SNE (Xu et al., 2019), to train a ResNet50

model (He et al., 2016) on the Office-Home dataset (Venkateswara et al., 2017) with 3022 samples of the source data and 338 samples a target data for 300 epochs takes  $> 18$  hours using a GTX 1080 Ti GPU. To put that in perspective, training a ResNet50 model on the larger CIFAR100 dataset with 45000 training samples for 300 epochs on the same machine for standard cross-entropy loss takes only 14 hours. The increase in training time not only increases energy consumption and CO2 emissions but also restricts the use of SDA methods in resource-constrained environments. We address this problem by seeking an answer to the following question: Can we improve the efficiency of SDA methods by training on subsets of source data to ensure faster adaptation and reduction in training time?

To this end, we propose ORIENT, a subset selection framework that utilizes Submodular Mutual Information (SMI) measures (Iyer et al., 2022; Gupta and Levin, 2020) to select subsets of source data that are similar to target data for training. Our key insight is that by training the model on data points that are similar to the target data, we can speed up the training. The ORIENT framework, illustrated in Figure 10.1, complements existing SDA methods and can be effectively combined with any of them, enabling us to achieve a reduction in training times, energy costs, and CO2 emissions.

The key contributions of this chapter include (1) ORIENT, an SMI-based subset selection framework for efficient and effective supervised domain adaptation. (2) In Section 10.5, we illustrate that ORIENT unifies three existing subset selection methods: CRAIG (Mirza-soleiman et al., 2020), GLISTER (Killamsetty et al., 2021), and GRADMATCH (Killamsetty et al., 2021). Specifically, these methods fit into ORIENT’s framework when used with a held-out validation set from the target domain. (3) We empirically show the effectiveness of ORIENT when used with existing SDA methods like CCSA (Motiian et al., 2017),  $d$ -SNE (Xu et al., 2019), and Standard Cross-entropy training on two real-world datasets: Office-31 (Saenko et al., 2010) and Office-Home (Venkateswara et al., 2017). Our experiments also demonstrate that a model learns better class discrimination features with ORIENT, resulting in better classification performance.

The rest of the chapter is organized as follows. Section 10.2 discusses the related work for SDA and subset selection. Section 10.3 reviews the required preliminaries for the proposed ORIENT pipeline. Then, Section 10.4 describes the proposed method. Section 10.5 discusses the connection of Orient with outer subset selection approaches. Section 10.6 presents the empirical evaluation of the proposed method. Finally, Section 10.7 concludes the chapter.

## 10.2 Related work

### 10.2.1 SDA

When the labeled training data is scarce for a domain  $\tau$  (or a task), a powerful way to boost performance is by pretraining a model on a related domain and fine-tuning it in the target domain (Girshick et al., 2014; Yosinski et al., 2014; Chu et al., 2016). SDA techniques have been investigated for various distribution shift settings where the target domain ( $\tau^t = \{X^t, Y^t\}$ ) has different marginal or conditional distributions than the source domain ( $\tau^s = \{X^s, Y^t\}$ ). In this work, we consider *covariate shift* (Shimodaira, 2000), wherein the marginal distribution is different (i.e.  $P(X^s) \neq P(X^t)$ ), but the conditional distribution remains the same (i.e.  $P(Y^s|X^s) = P(Y^t|X^t) = P(Y|X)$ ). A notable body of work for covariate shift SDA has focused on identifying a latent feature space that is domain-invariant (Motiian et al., 2017; Xu et al., 2019; Morsing et al., 2020; Hedegaard et al., 2021). This line of work train a network consisting of a feature extractor and a classifier end-to-end. The feature extractor learns a non-linear transformation of the samples from different domains to a shared latent space, and the classifier learns to assign class labels. Different researchers have proposed different optimization objectives and loss functions to encourage domain confusion and class separability.

Tzeng et al. (2015) designed a domain-confusion loss to optimize for domain invariance and match the distribution over classes in the source domain to the soft label in the target

domain. Motiian et al. (2017) propose a classification and contrastive semantic alignment loss (CCSA). CCSA uses contrastive loss with Siamese Network to encourage the same class samples from different distributions to be nearby in the latent space (semantic alignment loss), and the different class samples from different distributions to be far apart (separation loss). Few-shot adversarial domain adaptation (FADA) (Motiian et al., 2017) proposes to learn similar latent space with adversarial training. Domain-adaptation using stochastic neighborhood embedding ( $d$ -SNE) (Xu et al., 2019) proposes to maximize the smallest distance between the samples of different classes and minimizes the largest distance between the samples of the same class while being domain invariant. Another approach, Second- or Higher-order Transfer of Knowledge (So-HoT) (Koniusz et al., 2017), aims to align the within-class scatters and maintain separation of between-class scatters. Finally, Morsing et al. (2020) propose domain adaptation using graph embedding (DAGE) and show that CCSA and  $d$ -SNE can also be expressed as graph embedding methods (Hedegaard et al., 2021).

### 10.2.2 Subset selection methods

Submodular function (Lovász, 1982; Fujishige, 2005) is one of the effective approaches of data subset selection which is used in a variety of applications (Bilmes, 2022). For example, it has been employed for efficient training in speech recognition (Wei et al., 2014,?), machine translation (Kirchhoff and Bilmes, 2014), computer vision (Kaushal et al., 2019), supervised classification (Mirzasoleiman et al., 2020; Killamsetty et al., 2021,?), semi-supervised classification (Killamsetty et al., 2021), active learning methods (Wei et al., 2015; Sener and Savarese, 2018; Ash et al., 2020; Killamsetty et al., 2021), and hyper-parameter optimization (Killamsetty et al., 2022). Another widely used method for selecting subsets is coreset construction. The coresets (Feldman, 2020) are weighted subsets of the data that approximate the semantic characteristics of the entire dataset (e.g., loss, marginal probability, etc.). A number of recent coreset selection-based methods (Mirzasoleiman et al., 2020; Killamsetty

et al., 2021) have demonstrated the promise to efficiently and robustly train deep models. Coresets are used for other deep learning applications like continual learning, active learning, and data summarization (Borsos et al., 2021; Tiwari et al., 2022).

Subset selection plays a significant role in robust learning under realistic scenarios, such as imbalances or rare classes, out-of-distribution(OOD) data, or redundancy. Axelrod et al. (2011) propose a simple cross-entropy-based data selection method for effective domain adaptation in statistical machine learning. Kothawade et al. (2021) employed SMI measures for active learning in realistic scenarios tackling imbalance, OOD, and redundancy. PRISM (Kothawade et al., 2022) employed SMI measures for targeted data summarization. GLISTER (Killamsetty et al., 2021) and GRADMATCH (Killamsetty et al., 2021) show that using a clean held-out validation set with SMI mitigates the label noise and class imbalance in the training set. Mirzasoleiman et al. (2020) showed the data samples with clean labels cluster together, whereas the ones with noisy labels spread out in the gradient space, thus making k-medoid clustering an effective method for reducing noise in labeled data. Furthermore, Killamsetty et al. (2021) tackles OOD and imbalance in the unlabeled data in semi-supervised learning setting through data subset selection. We show how these previous subset selection strategies including a modified version of CRAIG (Mirzasoleiman et al., 2020) were also using instantiations of SMI measures in Section 10.5.

### 10.3 Preliminaries

In this section, we introduce submodular functions, SMIs, and SDA loss functions.

#### 10.3.1 Submodular functions

Let  $D$  be a set of  $n$  data points  $D = \{1, \dots, n\}$  and  $f : 2^D \rightarrow \mathbb{R}$  be a set function returning real-value for any subset of set  $D$ . A function  $f$  is *submodular* (Fujishige, 2005) if  $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$ ,  $\forall x \in D$ ,  $\forall A \subseteq B \subseteq D$  and  $x \notin B$ . Monotone Submodular

functions, when maximized for a cardinality constraint using a simple greedy algorithm, admit a constant factor of  $1 - \frac{1}{e}$ . This can be done in near-linear time using a stochastic-greedy algorithm.

### 10.3.2 Submodular mutual information

Submodular mutual information (SMI) between two sets  $A, B \subseteq D$ , *instantiated* with a submodular function  $f$ , is defined as  $I_f(A; B) = f(A) + f(B) - f(A \cup B)$  (Iyer et al., 2022). In this work, we aim to select data points  $A \subset D$  that maximize the SMI between  $A$  and  $B$ . Intuitively, this allows the selection of data points that are *similar* to  $B$  while being diverse.

### 10.3.3 SDA loss functions

For supervised domain adaptation, specialized loss functions are introduced as described in the previous section. Here we highlight two state-of-the-art domain adaptation loss functions, CCSA and  $d$ -SNE, which are used in our experiments.

In CCSA, the classifier is modeled as a composition of two functions— $h \circ g$ . Here,  $g : X \rightarrow Z$  is a feature extractor that transforms the input from the feature space  $X$  to an embedding space  $Z$ , and  $h : Z \rightarrow Y$  is a predictor function. The CCSA loss for supervised domain adaptation is defined as,

$$\mathcal{L}_{CCSA}(h \circ g) = \mathcal{L}_{CE}(h \circ g) + \mathcal{L}_{SA}(g) + \mathcal{L}_S(g),$$

where  $\mathcal{L}_{CE}(h \circ g)$  is the *cross-entropy loss* for multi-class classification,  $\mathcal{L}_{SA}$  is a *semantic alignment loss* encouraging the samples from different domains but the same label to map nearby in the embedding space, and  $\mathcal{L}_S$  is a *separation loss* encouraging the samples from different domains and different labels to map far away in the embedding space.

The  $d$ -SNE loss function for SDA is defined as

$$\mathcal{L}_{d\text{-SNE}}(h \circ g) = \tilde{\mathcal{L}}(g) + \alpha \mathcal{L}_{CE}^s(h \circ g) + \beta \mathcal{L}_{CE}^t(h \circ g),$$

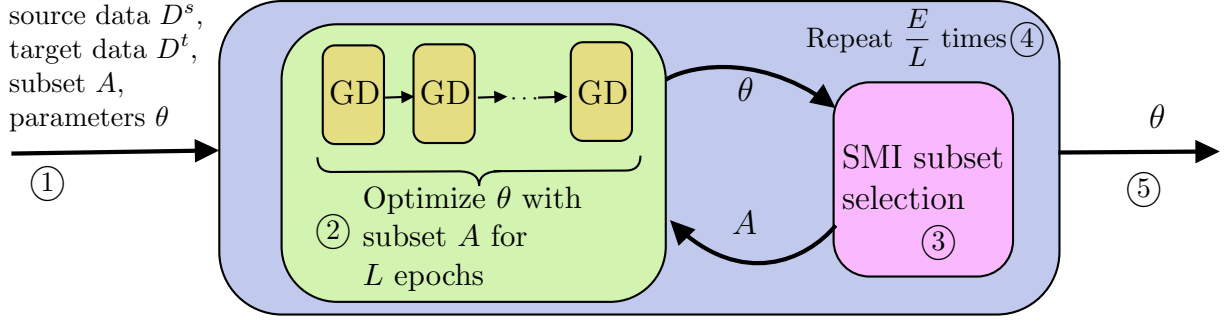


Figure 10.1: Illustration of ORIENT framework. ① Given the target data  $D^t$  and source data  $D^s$ , a subset  $A \subseteq D^s$  and model parameters  $\theta$  are randomly initialized. ② Model parameters  $\theta$  are optimized for  $L$  epochs on the subset  $A$  with any gradient descent (GD) based method. ③ Subset  $A$  is updated using the SMI measure and current model parameters  $\theta$  after every  $L^{th}$  epoch. ④ Model is trained for  $E$  epochs, with subset selection after every  $L$  interval. ⑤ The final model parameters are returned after  $E$  epochs.

where  $\mathcal{L}_{CE}^s$  and  $\mathcal{L}_{CE}^t$  are the cross-entropy loss on the source and the target domain examples, respectively, and  $\tilde{\mathcal{L}}$  minimizes the largest distance between the samples from different domains with the same label and maximizes the smallest distance between the samples from a different domain with a different label in the embedding space. SDA loss functions are further elaborated in Appendix D.3.

#### 10.4 Orient

For our SDA setting, we assume a small amount of labeled data (as less as 2 examples per class) is available from the target domain  $\tau^t$  and sufficient labeled data is available from the source domain  $\tau^s$ . We denote the source dataset as  $D^s = (x_i, y_i)_{i=1}^{|D^s|}$  and the target dataset as  $D^t = (x_i, y_i)_{i=1}^{|D^t|}$ . The source dataset is used for training the model  $\mathcal{M}$  using an SDA loss function  $\mathcal{L}$ . We will introduce other necessary notations in the remainder of the chapter, as necessary.

First, we extend the SMI measure to incorporate data subsets from different domains. For a source domain subset  $A \subseteq D^s$  and the target domain data set  $D^t$ , we denote the

Table 10.1: Instantiations of submodular mutual information functions.

Name	$f(A)$	$I_f(A; D^t)$
FLMI	$\sum_{i \in D^t} \max_{j \in A} S_{ij}$	$\sum_{i \in D^t} \max_{j \in A} S_{ij} + \eta \sum_{i \in A} \max_{j \in D^t} S_{ij}$
LOGDETM	$\log \det(S_A)$	$\log \det(S_A) - \log \det(S_A - \eta^2 S_{A,D^t} S_{D^t}^{-1} S_{A,D^t}^T)$
GCMI	$\sum_{i \in A, j \in D^s} S_{ij} - \lambda \sum_{i,j \in A} S_{ij}$	$2\lambda \sum_{i \in A, j \in D^t} S_{ij}$
COM	Appendix Eq. D.1	$\eta \sum_{i \in A} \psi(\sum_{j \in D^t} S_{ij}) + \sum_{j \in D^t} \psi(\sum_{i \in A} S_{ij})$

SMI measure as  $I_f(A; D^t)$ . Essentially, assuming that both the domains form a set  $\mathcal{D}$ , and  $A \subseteq D^s \subset \mathcal{D}$  and  $D^t \subset \mathcal{D}$ . With  $S$  denoting the similarity matrix defined over a subset  $A$  and  $D^t$ , Table 10.1 presents the mathematical expression of the SMI functions for different instantiations of submodular functions ( $f$ ) (Kothawade et al., 2022). We defer the readers to Kothawade et al. (2022) for more details. Note that the FLMI, GCMI, and COM only need the pairwise similarities of the data points in  $A$  and  $D^t$ . Consequently, the similarity kernel is of size  $|A| \times |D^t|$ , making it very efficient to optimize.

For efficient SDA, we want to select a subset  $A$  of the source domain  $\tau^s$  such that training a model on  $A$  results in a classifier that is proficient on the target domain  $\tau^t$ . To this effect, we use the gradients  $\chi$  of the current model to represent the data points and use it to compute the similarity matrix. Specifically, we define the pairwise similarity between two data points as the cosine similarity between the gradients,

$$S_{ij} = \frac{\chi_i \chi_i}{|\chi_i| |\chi_j|} \quad (10.1)$$

$$\text{where, } \chi_i = \nabla_{\theta} \mathcal{L}(x_i, y_i) \text{ and } \chi_j = \nabla_{\theta} \mathcal{L}(x_j, y_j)$$

Given a set size  $b$ , we select a *diverse* subset  $A$  of the source data  $D^s$  that is *similar* to the target data  $D^t$  by maximizing the following SMI measure,

$$\arg \max_{A \subseteq D^s, |A| \leq b} I_f(A; D^t) \quad (10.2)$$

$I_f(A; D^t)$  is an instance of cardinality-constrained monotone submodular maximization for all SMI functions except for LogDetMI. Therefore, we can achieve a  $1 - \frac{1}{e}$  approximation using the lazy greedy algorithm (Minoux, 1978). Even though LogDetMI is not submodular, previous works (Kothawade et al., 2022, 2021) have reported good empirical performance using the lazy greedy algorithm for maximization of the LogDetMI function. Following the footsteps of Kothawade et al. (2022, 2021), we also use the lazy greedy algorithm to maximize the LogDetMI function. However, as the number of parameters in the deep learning model can be extremely high, our gradients  $\chi$  can be very high dimensional. Following the success of targeted subset selection approaches, GLISTER (Killamsetty et al., 2021), SIMILAR (Kothawade et al., 2021), CRAIG (Mirzasoleiman et al., 2020) and BADGE (Ash et al., 2020), we circumvent this problem by using last-layer gradient approximations to represent the data point in the similarity matrix  $S$ . Further, we select a new subset every  $L$  epochs as the model is trained, and the subsets chosen are adapted accordingly.

#### 10.4.1 Algorithm

We present the complete training procedure of ORIENT in Algorithm 5. We first randomly initialize the training subset  $A$  and the model parameters  $\theta$  in **line 1** and **2**, respectively. For each epoch, we update the model parameters by optimizing a gradient-descent (GD) based loss  $\mathcal{L}$  on the current training subset  $A$  in **line 4**. After a fixed interval of  $L$  epochs, we update the subset  $A$  (**line 5–11**). To do this, we compute the gradients  $\chi^s$  and  $\chi^t$  for all the datapoints in  $D^s$  and  $D^t$ , in **line 6** and **7**, respectively. Next, we use the gradients to compute the similarity matrix  $S$  in **line 8**, as described in Equation 10.1. In **line 9** we instantiate the SMI function  $I_f$  with  $S$  and update the subset  $A$  in **line 10** using Equation 10.2. Our implementation is made available online<sup>1</sup>.

---

<sup>1</sup><https://github.com/athresh/orient>

---

**Algorithm 5** ORIENT

---

**Input:** source domain data  $D^s$ , query data  $D^t$ , batch size  $b$ , total epochs  $E$ , subset selection interval  $L$ , SMI function  $I_f$ .

**Output:** Final model parameters  $\theta$

```
1:  $A \leftarrow \text{RandomSubset}(D^s)$  ▷ Randomly initialize a subset
2:  $\theta$  ▷ initial model parameters
3: for epoch  $e$  in  $E$  do ▷ for each epoch
4:    $\theta \leftarrow \text{mini-batch-gd}(\theta, \mathcal{L}(A))$  ▷ mini-batch gradient descent
5:   if  $e \bmod L == 0$  then ▷ perform subset selection every  $L$  epochs
6:      $\chi^s \leftarrow \nabla_{\theta} \mathcal{L}(D^s)$  ▷ compute gradients for source data
7:      $\chi^t \leftarrow \nabla_{\theta} \mathcal{L}(D^t)$  ▷ compute gradients for query data
8:      $S \leftarrow \text{SIMILARITY}(\chi^s, \chi^t)$  ▷ compute the similarity matrix
9:     Instantiate  $I_f$  with  $S$ 
10:     $A \leftarrow \arg \max_{A \subseteq D^s, |A| \leq b} I_f(A; D^t)$  ▷ update subset  $A$ 
11:   end if
12: end for
```

---

Figure 10.1 illustrates the ORIENT workflow. The general framework of ORIENT can incorporate any gradient descent (GD) based SDA technique. It can train any model with loss function  $\mathcal{L}$  by using a subset of the source data, selected every  $L$  epochs. In our experiments, we demonstrate the effectiveness of our method in conjunction with standard cross-entropy loss as well as two state-of-the-art domain adaptation losses, CCSA (Motiian et al., 2017) and d-SNE (Xu et al., 2019).

### 10.5 Connections to previous work

ORIENT generalizes three approaches, GLISTER (Killamsetty et al., 2021), GRADMATCH (Killamsetty et al., 2021), and a slightly adapted version of CRAIG (Mirzasoleiman et al., 2020), which were initially proposed for efficient and robust subset selection. Specifically, these subset selection approaches maximize a submodular function corresponding to a differ-

ent instantiation of SMI functions for subset selection. The following theorems summarize the connection of SMI functions with these subset selection approaches.

**Theorem 3.** *When the outer level loss of the discrete bi-level optimization problem of GLISTER is hinge loss, logistic loss, and perceptron loss, then the optimization problem becomes an instance of Concave over Modular (COM) SMI function.*

**Theorem 4.** *When the optimization problem of GRADMATCH is used to match gradients of a held-out validation set, it becomes an instance of the summation of GCMI and a diversity function.*

**Theorem 5.** *When the optimization problem of CRAIG is adapted to match gradients of a held-out validation set, it becomes an instance of the FLMI function with  $\eta = 0$ .*

Proofs of these theorems are presented in Appendix D.2. In conclusion, SMI measures have been used in previous subset selection strategies for robust learning to deal with the class imbalance and noisy labels in training data sets and achieved comparable performance to current state-of-the-art robust supervised learning approaches.

## 10.6 Experimental evaluation

Our experiments explicitly aim at answering the following questions,

- Q1.** Does the proposed ORIENT approach substantially reduce the training time while maintaining comparable performance to training on complete source dataset?
- Q2.** Can the proposed ORIENT approach augment the existing domain adaptation approaches?

We evaluate our proposed approach on two domain adaptation datasets: Office-31 (Saenko et al., 2010) and Office-Home (Venkateswara et al., 2017). Office-31 dataset consists of 4110

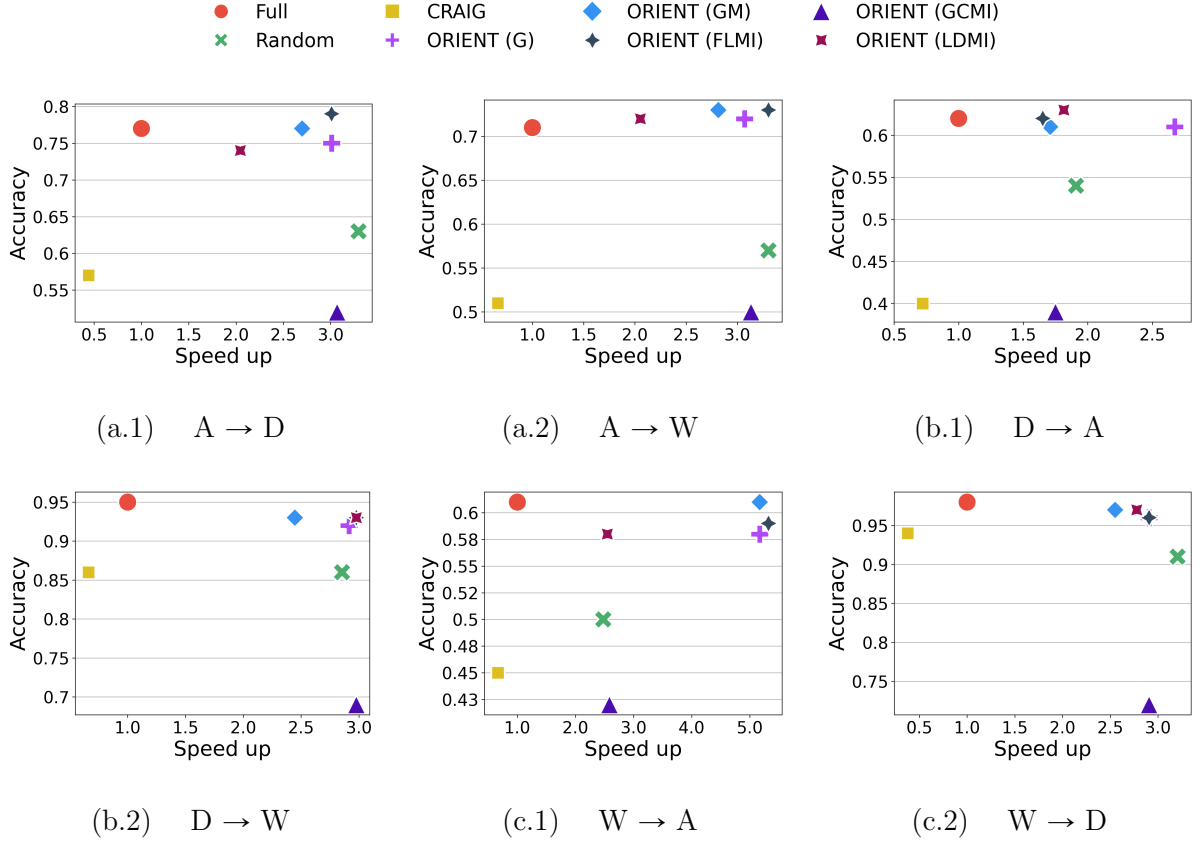


Figure 10.2: Speed up vs prediction accuracy on three domains of Office31 dataset: Amazon (A), DSLR (D), and Webcam (W).  $X$ -axis represents the speed up by the model, i.e. the ratio of the time taken to train on complete source dataset (Full) to the time taken by the model.  $Y$ -axis represents the prediction accuracy of the model on the target domain.

images of 31 object categories from three different domains: Amazon (A), DSLR (D), and Webcam (W). For our experiments, we used all the images from the source domain in the training set ( $D^s$ ) and two examples of each category in the query set ( $D^t$ ). Office-Home dataset consists of 10812 images of 65 object categories from four different domains: Art (A), Clipart (C), Product (P), and Real-world (R). Here, the target data ( $D^t$ ) consists of about 20% of the images from the target domain. We use  $L = 20$ , that is, we sample the subset after every 20 epochs and use  $b = 0.3\%$ , that is, we sample 30% of the source domain for training.

Table 10.2: Test accuracy for office-31 with SDA methods

		A $\rightarrow$ D	A $\rightarrow$ W	D $\rightarrow$ A	D $\rightarrow$ W	W $\rightarrow$ A	W $\rightarrow$ D
CCSA	Full	0.78	0.72	0.55	<b>0.93</b>	0.55	<b>0.97</b>
	Random	0.76	0.72	0.54	0.81	0.54	0.92
	ORIENT	0.77	<b>0.76</b>	0.55	0.89	0.55	<b>0.96</b>
d-SNE	Full	0.77	0.69	0.53	<b>0.93</b>	0.54	<b>0.98</b>
	Random	0.76	0.68	0.53	0.86	0.53	0.94
	ORIENT	0.78	0.71	<b>0.55</b>	0.90	0.56	<b>0.97</b>

Table 10.3: Test accuracy for Office-Home with SDA methods

		R $\rightarrow$ P	R $\rightarrow$ C	P $\rightarrow$ R	P $\rightarrow$ C	C $\rightarrow$ R	C $\rightarrow$ P	A $\rightarrow$ P	A $\rightarrow$ R	A $\rightarrow$ C	R $\rightarrow$ A	P $\rightarrow$ A	C $\rightarrow$ A
CCSA	Full	0.74	<b>0.55</b>	0.62	0.46	0.56	0.67	0.70	<b>0.64</b>	0.48	0.57	<b>0.49</b>	0.41
	Random	0.71	0.47	0.62	0.45	0.54	0.66	0.69	0.57	0.48	0.5	0.47	0.45
	ORIENT	<b>0.78</b>	<b>0.54</b>	<b>0.65</b>	<b>0.5</b>	<b>0.61</b>	<b>0.71</b>	0.71	<b>0.65</b>	<b>0.51</b>	<b>0.59</b>	<b>0.54</b>	<b>0.47</b>
d-SNE	Full	0.77	<b>0.53</b>	<b>0.62</b>	<b>0.50</b>	<b>0.60</b>	0.71	<b>0.72</b>	<b>0.63</b>	0.49	<b>0.52</b>	0.44	0.40
	Random	0.75	0.50	0.60	0.45	0.57	0.69	0.68	0.59	0.49	0.46	0.43	0.40
	ORIENT	0.77	<b>0.52</b>	<b>0.63</b>	<b>0.50</b>	<b>0.60</b>	0.71	<b>0.71</b>	0.61	0.51	<b>0.52</b>	0.44	0.42

To answer the first question, we use ORIENT to train a ResNet50 architecture using the stochastic gradient descent (SGD) algorithm with the momentum of 0.9 and the weight decay ratio of  $5e4$ . We compare our approach against three baselines: **Full**—training on the source data and the target data, i.e.  $D^s \cup D^t$ ; **Random**—a random subset of the source dataset and the target set, i.e.  $R \cup D^t, R = \text{RandomSubset}(D^s)$ ; and **CRAIG**—a coreset of the source dataset, i.e.  $\text{coreset}(D^t)$ , without any information of the target dataset. We use standard categorical cross-entropy loss function (for multi-class classification) and five different instantiations of the submodular mutual information function: Facility Location Mutual Information (**ORIENT (FLMI)**), Graph Cut Mutual Information (**ORIENT (GCMI)**), Log Determinant Mutual Information (**ORIENT (LDMI)**), GLISTER (**ORIENT (G)**), and GradMatch (**ORIENT (GM)**).

Figure 10.2 presents the scatter plot of the speed up against the prediction accuracy of the Office-31 dataset using cross-entropy loss. In all our experiments, we use a 0.3 fraction of the source data as the subset and see  $> 2.5\times$  speed up. Figure 10.3 presents the convergence

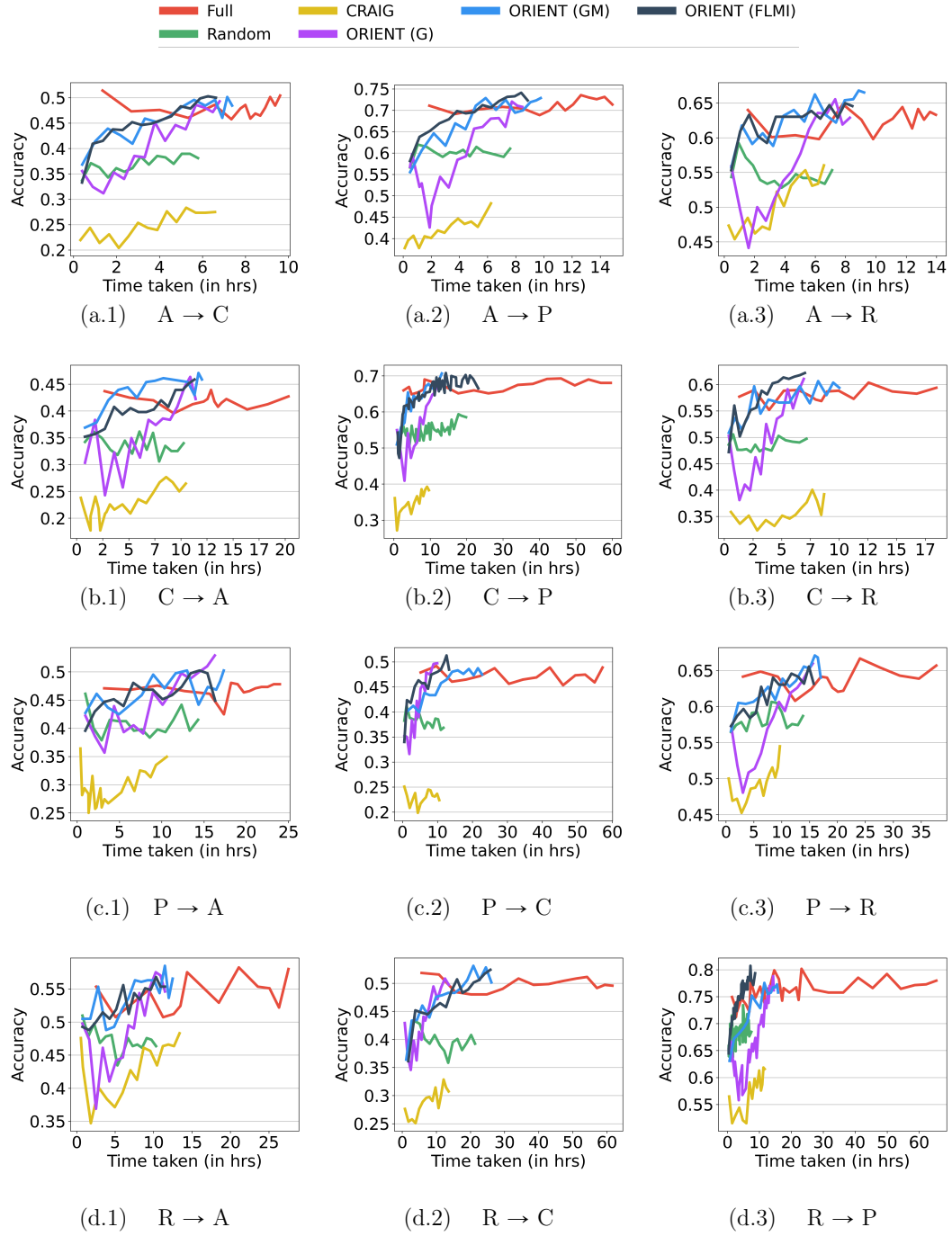


Figure 10.3: Convergence curves on four domains of Office-Home dataset: Art (A), Clipart (C), Product (P), and Real World (R). X-axis presents the training time in hours and Y-axis presents the prediction accuracy on the target domain.

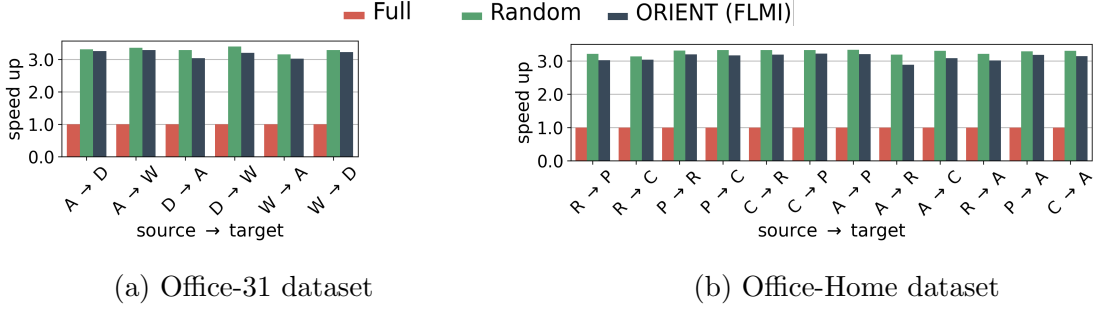


Figure 10.4: Speed up achieved by combining d-SNE with ORIENT

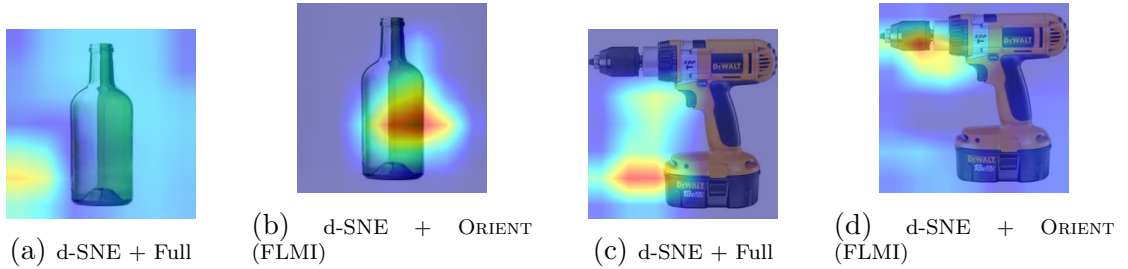


Figure 10.5: GradCam (Selvaraju et al., 2017) activation maps of the models learned using d-SNE + Full and d-SNE + ORIENT (FLMI) on the Office-Home dataset with “Product” as the source domain and “Real World” as the target domain. As evidenced by class activation maps, the ORIENT framework enabled the model to learn more effective class discriminative features than Full data training.

curves of the Office-Home dataset using cross-entropy loss. It is evident from these charts that our proposed approach ORIENT can achieve better or comparable performance to the Full training in significantly less amount of time, indicating better efficiency. This analysis helps us answer our Q1. ORIENT reduces the training time substantially, and the speed up achieved is reciprocal to the fraction of the subset used, without trading the performance. Rather, in some combinations, we observe that ORIENT outperforms Full.

Figure 10.4 presents the bar plots of the speed up when ORIENT(FLMI) is used in conjunction with *d*-SNE loss function on Office-31 and Office-Home datasets. Here, we see a consistent  $3\times$  speed up as compared to full model training with *d*-SNE loss. These bar plots demonstrate that our proposed approach ORIENT can augment existing domain adaptation approaches and substantially reduce the training times. Additionally, Tables 10.2

and 10.3 present the test accuracy while using  $d$ -SNE and CCSA loss in conjunction with ORIENT(FLMI) on Office-31 and Office-Home datasets, respectively. It’s evident that augmenting existing domain adaptation approaches with ORIENT achieves better or comparable performance to the Full training. These two observations help us answer our Q2 in the affirmative. Precise values of prediction accuracy and training time for all the experiments are provided in Appendix D.5. Further, in Appendix D.5.2 and Appendix D.5.3, we analyze the effect of different subset sizes and subset sampling frequencies.

Figure 10.5 presents the GradCam (Selvaraju et al., 2017) class-activation maps of trained models on the Office-Home dataset ( $P \rightarrow R$  setting) using  $d$ -SNE loss for both Full and ORIENT (FLMI). These activation maps show that the model trained with the ORIENT framework learned better class discrimination features than the model trained with Full. This might explain why the ORIENT framework performs better sometimes than Full.

### 10.6.1 Comparison of different SMI functions

Domain adaptation results on Office31 (Figure 10.2) and OfficeHome (Figure 10.3) datasets show that ORIENT(GCMI) performs suboptimally compared to ORIENT using other SMI functions in terms of target domain accuracy. Although ORIENT(LDMI) achieves reasonable target domain accuracy, it is computationally expensive and does not achieve the best performance-speedup trade-off. In comparison to the ORIENT using remaining SMI functions, i.e., ORIENT(GM), ORIENT(G), and ORIENT(FLMI), ORIENT(FLMI) consistently achieves the best performance versus speed-up trade-off. We further present synthetic experiments to provide intuitions on how different SMI functions selects data subsets in Appendix D.5.1

To summarize, our experiments on two adaptation datasets suggest that our proposed approach ORIENT can substantially reduce the training time while maintaining comparable performance to training on complete source data. Additionally, our experiments using  $d$ -SNE

and CCSA loss functions suggest that ORIENT can be used in conjunction with existing SDA methods to achieve significant speed ups in training time, while maintaining or improving the prediction accuracy. In particular, two instantiations of our proposed approach, ORIENT(FLMI) and ORIENT(GM) consistently achieve comparable or better performance compared to Full training while being  $\sim 3\times$  faster to train.

## 10.7 Summary

We introduce ORIENT, a subset selection framework based on SMI functions for supervised domain adaptation. The submodularity of SMI functions allows us to use scalable greedy algorithms to select the data subsets efficiently. In addition, we demonstrate how ORIENT is a unified framework that integrates previous approaches based on subset selection for robust learning. Empirically, we show that ORIENT is very effective for SDA. Specifically, it achieves  $\sim 3\times$  speed up over existing SDA approaches like  $d$ -SNE and CCSA while achieving comparable or better performance. Our findings confirm that ORIENT has a significant social impact by making existing SDA algorithms significantly faster and more energy-efficient, reducing CO2 emissions and energy consumption incurred during training, thus contributing to a Green AI (Schwartz et al., 2020). The main limitation of our work is that although ORIENT significantly reduces the training time, it requires more memory to store the similarity kernel required for subset selection. This makes running ORIENT harder on devices with low memory.

## CHAPTER 11

### CONCLUSION AND FUTURE WORK

For an effective and efficient adaptation of AI and ML approaches in real-world use cases, approaches must be able to *interact with, learn from, and teach* humans. This dissertation contributes to *interact with* and *learn from* humans part. We looked beyond learning from data and explored ways to leverage rich human knowledge to guide the learning procedure. We also proposed a system that can interact with humans in natural language.

In Part I (Chapter 3), we considered the challenge of learning from sparse and noisy data in the successful gradient-boosting framework. We proposed the Knowledge-intensive Gradient Boosting (KiGB) framework to use the qualitative influence information to improve prediction. KiGB uses the monotonicity information as soft constraints to gently nudge the leaf values in the direction of the influence. Our experiments show that KiGB with its domain knowledge is better than learning from data alone. In some cases, the knowledge does not help, but it never hurts. We specifically encountered two scenarios where the knowledge does not help. First, when the data conforms to the knowledge, that is, data is clean and never gives a contradictory signal to the model while training. Second, is when the knowledge is less relevant. That is when the monotonic influence is about a feature that is correlated with the target feature. In that case, the influent feature may or may not appear in the tree. Hence, the KiGB soft constraints do not influence the predictions. When the knowledge is indeed relevant, KiGB observes a jump start and better slope for learning, and more importantly, it also achieves a higher asymptote in performance. Our empirical evaluations show that knowledge is most helpful when the data is sparse or noisy. A compelling future research direction would be to investigate how to estimate the quality of knowledge and evaluate the impact of knowledge quality on learning.

Then we consider the challenge of learning to generalize across multiple tasks and objects in sequential decision making. We address this challenge by proposing the RePreL frame-

work and its extension in Part II (Chapter 4–6). RePreL framework integrates relational planning and reinforcement learning. It takes inspiration from human’s ability to generalize by identifying compositionality and leveraging task-specific abstract representations. In the RePreL framework, we procc strives to incorporate such domain knowledge. RePreL uses a high-level domain description to decompose the task into subtasks and leverages task-specific influence information for generating abstract representations. Our empirical evaluations, on discrete, continuous, and hybrid domains (in Chapter 4, 5, and 6, respectively), indicate that the RePreL framework not only demonstrates a significant advantage in sample efficiency but also shows the ability to generalize across multiple objects. RePreL facilitates the effective transfer of skills across tasks and the batch algorithm can adapt to any off-policy deep RL algorithm. Although transferred policy in general showed a better slope on the learning curves, we see that transferring neural policies may be slow sometimes. This phenomenon of a trained neural network losing the ability to quickly fit a new function is called capacitive loss (Lyle et al., 2022; Igl et al., 2021; Ash et al., 2020). It would be interesting to study when a policy or skill transfers seamlessly and when it shows resilience to transfer. Further, the current RePreL framework assumes the influence information provided by humans to be accurate. To quantify the influence and measure its uncertainty is an important next step to ensure the robustness of RePreL. One follow-up research question that we are currently pursuing is *can the RePreL framework be used in collaborative multi-agent environments*.

In Part III, we consider the challenging task of human-machine collaborative problem-solving in a Minecraft building task. This task is challenging as it requires sophisticated language understanding, contextual understanding, bi-directional communication, and composable vocabulary. We present an integrated system, Lara (see Chapter 7), that uses a rich natural language instruction from humans and maintains a first-order logic representation of knowledge that allows effective generalization while not sacrificing efficient reasoning. Lara uses an ILP engine to learn hierarchical, generalized concepts from a single example by

asking questions to humans to reduce uncertainty. Our system demonstrates that current advancements in NLP, planning, and ILP can be leveraged for the challenging collaborative problem-solving task. In our study, we used two types of NLP parsers a template-based based and an AMR-based parser. While the template-based parser was quite fast, it was limited in the instructions it could handle. We hoped for the AMR-based parser to support free-form sentences, however, found logic representation generated from AMR did not match the logic representation expected by the planner. In the current work, we ended up using rules to map the logic representation from AMR to the expected logic representation of the planner. However, bridging the gap between logic representations and extending planners to work with multiple logic representations can be an interesting future direction. Lara uses guided one-shot concept induction (GOCI) framework. While GOCI is shown to be successful in some important domains, it is limited. The GOCI framework uses a normalized compression distance (NCD) as a metric to compute the similarity between concepts. However, NCD is sensitive to the size of the plan and the characters used to represent the action. Devising an appropriate metric to compute concept similarity is an interesting future direction. Finally, with the latest advancements in the sequence-to-sequence models and the success of GPT architecture in natural language generation, it would be interesting to consider how these models perform bi-directional communication for collaborative problem-solving.

In Part IV, we considered the challenging task of learning rules in domains with heterogeneous objects. We looked at current neurosymbolic approaches and the various biases infused in their architectures. We propose two approaches in Chapter 8 to incorporate type-information as bias and evaluate these approaches against a current state-of-the-art. We found that while masked shows significant improvement, learning typed representations do not improve the rules. We plan to further investigate a few variants of similarity measures.

Lastly, in Part V, we looked at scenarios where the domain knowledge is not readily available. We specifically explore two directions in this scenario. One is to extract knowledge

from data, and another is to use data from related domains. In Chapter 9, we considered extracting the qualitative influence information from the data. We proposed QuaKE framework that extracts influence information by estimating joint distribution. We conclude that learning joint distribution using a causal model is much more effective and can provide rich insights into the qualitative influence between variables. Extending QuaKE framework—to learn the influence between random variables in a sequential decision making systems—would allow us to learn the D-FOCI statements used in Chapter 4. We leave this for future research. In Chapter 10, we consider using supervised domain adaptation techniques to train the model on related domains. We propose ORIENT framework that uses submodular mutual information measure for subset selection. ORIENT with the existing SDA approach is significantly faster and more efficient. However, the ORIENT approach computes the similarity kernel over the complete dataset. This might be limiting as it requires more memory.

In summary, this dissertation explored ways of incorporating human knowledge into different AI and ML approaches. We observe that using domain knowledge improves sample efficiency across the board and, more importantly, can improve the performance and generalization abilities of the model.

## **PART VI**

## **APPENDIX**

## APPENDIX A

### KNOWLEDGE-INTENSIVE GRADIENT BOOSTING

#### A.1 Why gradient boosted trees?

While different machine learning approaches have shown success for different problems, we specifically decided to incorporate qualitative knowledge in gradient-boosted trees. We made this decision as in our target domain of logistics the gradient-boosted trees consistently outperformed other models. This section of the appendix provides details of our analysis for making this decision.

We obtained a shipment price prediction dataset from a logistics company, Turvo<sup>1</sup>. For 20 different combinations of relevant feature sets, we analyzed the prominent implementations of the standard machine learning models. Specifically, we compared the Sklearn (Pedregosa et al., 2011)’s implementation of linear support vector machine (SVR-L), linear regression with L1 prior as regularizer (Lasso), support vector machine with radial basis function kernel (SVR-K), bagging ensemble of decision trees (Bagging), and gradient-boosted decision trees (GBT) and Tensorflow (Abadi et al., 2016)’s implementation of deep neural network regressor (DNN).

We used various domain-specific feature engineering techniques in this evaluation. The complete list of the 20 feature sets used is provided in Table A.1. Across all these feature sets, over 5 runs, we found that the gradient-boosted trees consistently outperformed the other models in terms of mean absolute percentage error (MAPE). The aggregated results are presented in Figure A.1.

---

<sup>1</sup>Turvo Inc. <https://turvo.com>

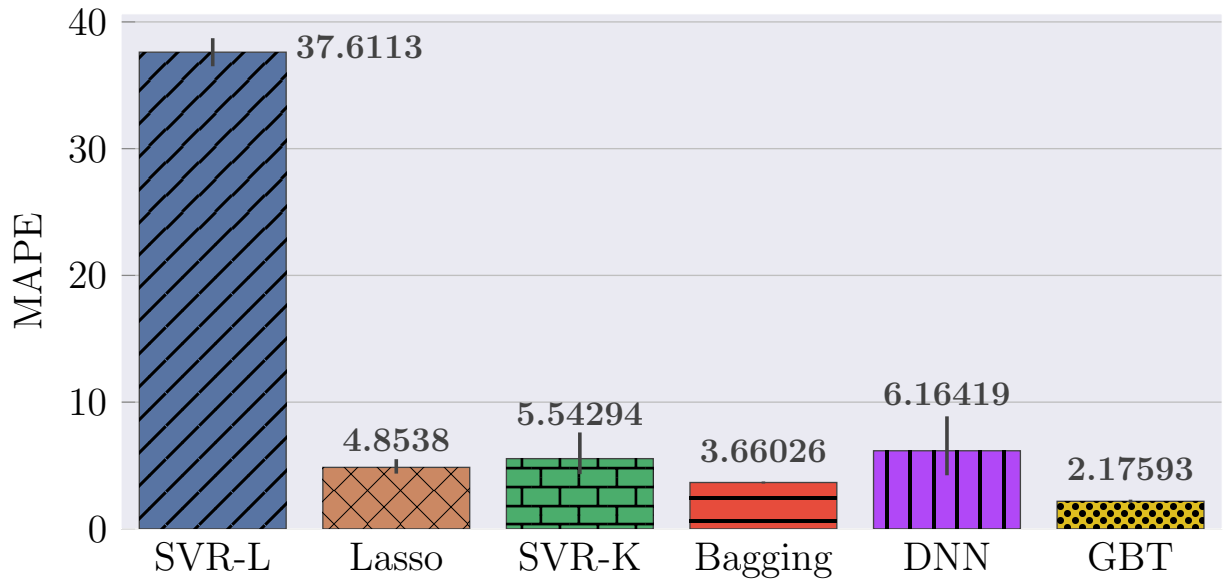


Figure A.1: Comparison of linear support vector regression (SVR-L), linear regressor with L1 prior as a regularizer (Lasso), support vector regressor with radial basis function kernel (SVR-K), bagging ensemble of decision trees (Bagging), deep neural network estimator (DNN), and gradient-boosted decision trees (GBT) for the task of predicting the shipment prices in Logistics dataset. Y-axis presents the mean absolute percentage error (MAPE) in prediction.

Table A.1: List of features used for the shipment price prediction task on logistics data.

	Features
1	equipment_temp, equipment_type, equipment_size, equipment_weight, lane_end_state, lane_start_state, lane_start, lane_end, miles, creation_month, creation_day, start_month, start_day, end_month, end_day, end_of_year, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, customer_id, carrier_id, national_average_fuel_price, average_fuel_price, fuel_surcharge, count_stops, count_items, count_customer_orders, interstate_shipment, round_trip, ltr_start_state, ltr_end_state, low_line_haul_rate, avg_line_haul_rate, high_line_haul_rate, fuel_surcharge_dat, new_year, cross_ratio, head_haul, back_haul, team_driver
2	lane_end_state, lane_start_state, lane_start, lane_end, miles, creation_month, creation_day, start_month, start_day, end_month, end_day, end_of_year, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, customer_id, carrier_id, national_average_fuel_price, average_fuel_price, fuel_surcharge
3	lane_start, lane_end, miles, creation_month, creation_day, start_month, start_day, end_month, end_day, end_of_year, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, customer_id, carrier_id, national_average_fuel_price, average_fuel_price, fuel_surcharge

Table A.1 continued

	Features
4	lane_start, lane_end, miles, end_of_year, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, customer_id, carrier_id, national_average_fuel_price, average_fuel_price, fuel_surcharge
5	lane_start, lane_end, miles, end_of_year, end_of_month, end_of_week, holiday, customer_id, carrier_id, national_average_fuel_price, average_fuel_price, fuel_surcharge
6	lane_start, lane_end, miles, end_of_year, end_of_month, end_of_week, holiday, customer_id, carrier_id, average_fuel_price, fuel_surcharge
7	lane_start, lane_end, miles, end_of_year, end_of_month, end_of_week, holiday, customer_id, carrier_id, average_fuel_price
8	lane_start, lane_end, miles, end_of_year, end_of_month, end_of_week, holiday, customer_id, carrier_id, average_fuel_price, round_trip, ltr_start_state, ltr_end_state, avg_line_haul_rate, new_year, cross_ratio, team_driver
9	lane_start, lane_end, miles, end_of_year, creation_month, start_month, end_of_month, end_of_week, holiday, carrier_id, average_fuel_price, round_trip, ltr_start_state, ltr_end_state, avg_line_haul_rate, new_year, cross_ratio, team_driver, customer_id
10	lane_start, lane_end, miles, end_of_year, creation_month, start_month, end_month, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, carrier_id, average_fuel_price, round_trip, ltr_start_state, ltr_end_state, avg_line_haul_rate, new_year, cross_ratio, team_driver, customer_id
11	lane_start, lane_end, miles, end_of_year, creation_month, start_month, end_month, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, carrier_id, average_fuel_price, round_trip, ltr_start_state, ltr_end_state, avg_line_haul_rate, new_year, cross_ratio, team_driver, customer_id, head_haul, back_haul
12	lane_start, lane_end, miles, end_of_year, creation_month, start_month, end_month, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, carrier_id, average_fuel_price, count_stops, round_trip, ltr_start_state, ltr_end_state, avg_line_haul_rate, new_year, cross_ratio, team_driver, customer_id, head_haul, back_haul
13	lane_start, lane_end, miles, end_of_year, end_of_month, end_of_week, holiday, customer_id, carrier_id, average_fuel_price, round_trip, ltr_start_state, ltr_end_state, avg_line_haul_rate, new_year, cross_ratio, team_driver, head_haul, back_haul
14	equipment_temp, equipment_type, equipment_size, equipment_weight, lane_end_state, lane_start_state, lane_start, lane_end, miles, creation_month, creation_day, start_month, start_day, end_month, end_day, end_of_year, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, customer_id, carrier_id, national_average_fuel_price, average_fuel_price, fuel_surcharge

Table A.1 continued

	Features
15	equipment_temp, equipment_type, equipment_size, equipment_weight, lane_end_state, lane_start_state, lane_start, lane_end, miles, creation_month, creation_day, start_month, start_day, end_month, end_day, end_of_year, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, national_average_fuel_price, average_fuel_price, fuel_surcharge, count_stops, count_items, interstate_shipment, round_trip, ltr_start_state, ltr_end_state, low_line_haul_rate, avg_line_haul_rate, high_line_haul_rate, fuel_surcharge_dat, new_year, cross_ratio, team_driver
16	equipment_temp, equipment_type, equipment_size, equipment_weight, lane_end_state, lane_start_state, lane_start, lane_end, miles, start_month, start_day, end_day, lead_time, transport_time, delivery_time, holiday, average_fuel_price, fuel_surcharge, count_stops, count_items, count_customer_orders, interstate_shipment, round_trip, low_line_haul_rate, avg_line_haul_rate, high_line_haul_rate, fuel_surcharge_dat, head_haul, back_haul
17	equipment_temp, equipment_type, equipment_size, equipment_weight, lane_end_state, lane_start_state, lane_start, lane_end, miles, start_month, start_day, end_month, end_day, end_of_year, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, average_fuel_price, fuel_surcharge, count_stops, count_items, count_customer_orders, interstate_shipment, round_trip, ltr_start_state, ltr_end_state, low_line_haul_rate, avg_line_haul_rate, high_line_haul_rate, fuel_surcharge_dat, head_haul, back_haul, team_driver
18	equipment_temp, equipment_type, equipment_size, equipment_weight, lane_end_state, lane_start_state, lane_start, lane_end, miles, creation_month, creation_day, start_month, start_day, end_month, end_day, end_of_year, end_of_month, end_of_week, lead_time, transport_time, delivery_time, holiday, national_average_fuel_price, average_fuel_price, fuel_surcharge, count_stops, count_items, count_customer_orders, interstate_shipment, round_trip, ltr_start_state, ltr_end_state, low_line_haul_rate, avg_line_haul_rate, high_line_haul_rate, fuel_surcharge_dat, new_year, cross_ratio, team_driver
19	miles
20	All

## APPENDIX B

### REPREL

#### B.1 Traditional relational RL

To compare the performance of RePreL against traditional relational RL (RRL), we persevered to implement two RRL approaches: Q-tree (Dzeroski et al., 2001) and Gradient Boosted Q-Learning (Das et al., 2020). The code implemented as part of this effort is made public at the following GitHub repository: <https://github.com/harshakokel/Relational-Q-Learning>. We tried to train an RRL policy for the extended taxi domain (see § 4.4) as well as the stack, unstack, on, and logistics domain from Das et al. (2020). While the Bellman error of the policies seemed to reduce with training, the success rate of these policies did not improve. This can perhaps be explained by the phenomenon studied by Fujimoto et al. (2022), that shows bellman errors are poor estimators for value functions.

Another observation worth noting is that the stack, unstack, on, and logistics domains have the same goal state for all the episodes. However, in the taxi domain, the goal state is different in every episode as the passenger destinations are sampled. Hence, the extended taxi domain is a little harder problem.

## APPENDIX C

### LARA

#### C.1 Dialogue manager

Dialogue manager in LARA maintains a fixed set of template sentences with slots shown below. As required the slots are populated and the natural language sentence generated is sent to the architect.

##### **Unknown shape**

I do not understand this shape. Try building it using tower or row.

Was the shape misspelled? May be try building a row

System does not support this shape. Do you wanna build a row instead?

##### **missing spatial relation**

(Can you describe—Could you tell me—Can you tell me) where the {new\_structure} is placed with respect to the {existing\_structure} (we just built)?

##### **missing dimension**

(What is the—What's the—Can you describe the—Could you tell me the) {missing\_dimension} of the {new\_structure}?

##### **Parse error for spatial relation**

Could not understand the spatial relationship. Can you say something like: The left-end of the row is on the east of the top-end of tower

Embarassed, I do not understand that.

**Parse error**

Sorry, I had trouble understanding that. Could you explain it differently?

Sorry, I don't understand. Can you try again?

Sorry, I'm having trouble understanding. Could you reword that?

**Planning error**

Sorry, I'm not able to do that. Could we try again?

Sorry, I'm not able to build that. Could you reword that?

Sorry, I can't do that. Could you explain it differently?

**Parse Clarification**

Sorry, I had trouble understanding that.

Sorry, I don't understand.

Sorry, I'm having trouble understanding.

**Initial Greeting**

Hi Architect, what are we building today?

I'm ready! What are we building?

Hello! What are we building?

Hello Architect, I'm ready!

**Next Greeting**

Hi Architect, what are we building today?

I'm ready! What are we building?

Hello! What are we building?

Hello Architect, I'm ready!

### Next Prompts

Okay, what's next?

Okay, now what?

What are we doing next?

## C.2 Background file

The complete list of predicates used in the FOL language can be found in the background file furnished below.

```
setParam: nodeSize=100.
setParam: loadAllBasicModes & false.

// Parts

// Shapes
mode: row(+Part).
mode: column(+Part).
mode: tower(+Part).
mode: square(+Part).
mode: rectangle(+Part).
mode: cube(+Part).
mode: cuboid(+Part).
```

```

mode: block(+Block).
mode: blockS(+Part).

// Dimensions
mode: width(+Part, #FloatPart).
mode: height(+Part, #FloatPart).
mode: length(+Part, #FloatPart).
mode: size(+Part, #FloatPart).

// Properties
mode: color(+Part, #ColorPart).
mode: spatial_rel(&rel, +Loc, +Loc).
mode: location(+Part).

// relation

mode: top_behind_left(+Part, -Block).
mode: top_left_behind(+Part, -Block).
mode: behind_top_left(+Part, -Block).
mode: behind_left_top(+Part, -Block).
mode: left_behind_top(+Part, -Block).
mode: left_top_behind(+Part, -Block).
mode: top_behind_right(+Part, -Block).
mode: top_right_behind(+Part, -Block).

```

```
mode: behind_top_right(+Block,-Block).
mode: behind_right_top(+Part,-Block).
mode: right_behind_top(+Part,-Block).
mode: right_top_behind(+Part,-Block).
mode: top_front_left(+Part,-Block).
mode: top_left_front(+Part,-Block).
mode: front_top_left(+Part,-Block).
mode: front_left_top(+Part,-Block).
mode: left_front_top(+Part,-Block).
mode: left_top_front(+Part,-Block).
mode: top_front_right(+Part,-Block).
mode: top_right_front(+Part,-Block).
mode: front_top_right(+Part,-Block).
mode: front_right_top(+Part,-Block).
mode: right_front_top(+Part,-Block).
mode: right_top_front(+Part,-Block).
mode: bottom_behind_left(+Part,-Block).
mode: bottom_left_behind(+Part,-Block).
mode: behind_bottom_left(+Part,-Block).
mode: behind_left_bottom(+Part,-Block).
mode: left_behind_bottom(+Part,-Block).
mode: left_bottom_behind(+Part,-Block).
mode: bottom_behind_right(+Part,-Block).
mode: bottom_right_behind(+Part,-Block).
mode: behind_bottom_right(+Part,-Block).
```

```
mode: behind_right_bottom(+Part,-Block).
mode: right_behind_bottom(+Part,-Block).
mode: right_bottom_behind(+Part,-Block).
mode: bottom_front_left(+Part,-Block).
mode: bottom_left_front(+Part,-Block).
mode: front_bottom_left(+Part,-Block).
mode: front_left_bottom(+Part,-Block).
mode: left_front_bottom(+Part,-Block).
mode: left_bottom_front(+Part,-Block).
mode: bottom_front_right(+Part,-Block).
mode: bottom_right_front(+Part,-Block).
mode: front_bottom_right(+Part,-Block).
mode: front_right_bottom(+Part,-Block).
mode: right_front_bottom(+Part,-Block).
mode: right_bottom_front(+Part,-Block).
mode: behind_left(+Part,-Block).
mode: left_behind(+Part,-Block).
mode: behind_right(+Part,-Block).
mode: right_behind(+Part,-Block).
mode: front_left(+Part,-Block).
mode: left_front(+Part,-Block).
mode: front_right(+Part,-Block).
mode: right_front(+Part,-Block).
mode: left_end(+Part,-Block).
mode: right_end(+Part,-Block).
```

```

mode: front_end(+Part,-Block).
mode: behind_end(+Part,-Block).
mode: top_end(+Part,-Block).
mode: bottom_end(+Part,-Block).
mode: block_location(+Block,-Loc).

// Bridgers

bridger: contains/2.
bridger: spatial_rel/3.

// Precomputes
mode: sameColor(+ColorShape,+ColorPart).
mode: sameSP(+FloatShape,+FloatPart).
mode: sameSS(+FloatShape,+FloatPart).
//mode: samePP(+FloatPart,+FloatPart).
mode: oneMoreSP(+FloatShape,+FloatPart).
mode: oneMorePS(+FloatPart,+FloatShape).
//mode: oneMorePP(+FloatPart,+FloatPart).
mode: oneMoreSS(+FloatShape,+FloatShape).

precompute: sameColor(X, Y) :- colorShape(Shape,X), color(Part,
    Y), X is Y.
precompute: sameSP(X, Y) :- heightShape(Shape,X), height(Part,Y
    ), sameAs(X, Y).

```

```

precompute: sameSP(X, Y) :- widthShape(Shape,X), width(Part,Y),
    sameAs(X, Y).
precompute: sameSP(X, Y) :- lengthShape(Shape,X), length(Part,Y
    ), sameAs(X, Y).
precompute: sameSP(X, Y) :- sizeShape(Shape,X), size(Part,Y),
    sameAs(X, Y).
precompute: sameSS(X, Y) :- heightShape(Shape,X), widthShape(
    Shape,Y), sameAs(X, Y).
precompute: sameSS(X, Y) :- heightShape(Shape,X), lengthShape(
    Shape,Y), sameAs(X, Y).
precompute: sameSS(X, Y) :- heightShape(Shape,X), sizeShape(
    Shape,Y), sameAs(X, Y).
precompute: sameSS(X, Y) :- widthShape(Shape,X), lengthShape(
    Shape,Y), sameAs(X, Y).
precompute: sameSS(X, Y) :- widthShape(Shape,X), sizeShape(
    Shape,Y), sameAs(X, Y).
precompute: sameSS(X, Y) :- lengthShape(Shape,X), sizeShape(
    Shape,Y), sameAs(X, Y).

//precompute: samePP(X, Y) :- height(Part1,X), width(Part2,Y),
    sameAs(X, Y).
//precompute: samePP(X, Y) :- height(Part1,X), width(Part2,Y),
    sameAs(X, Y).
//precompute: oneMorePP(X, Y) :- height(Part1,X), width(Part2,Y
    ), minus(X, Y, Z), Z is 1.

```

```

//precompute: oneMorePP(X, Y) :- height(Part1,X), height(Part2,
    Y), minus(X, Y, Z), Z is 1.
//precompute: oneMorePP(X, Y) :- width(Part1,X), height(Part2,Y
    ), minus(X, Y, Z), Z is 1.

precompute: oneMoreSS(X, Y) :- heightShape(Shape,X),
    heightShape(Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- heightShape(Shape,X), widthShape
    (Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- heightShape(Shape,X),
    lengthShape(Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- heightShape(Shape,X), sizeShape(
    Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- widthShape(Shape,X), heightShape
    (Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- widthShape(Shape,X), widthShape(
    Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- widthShape(Shape,X), lengthShape
    (Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- widthShape(Shape,X), sizeShape(
    Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- lengthShape(Shape,X),
    heightShape(Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- lengthShape(Shape,X), widthShape
    (Shape,Y), minus(X, Y, Z), Z is 1.

```

```

precompute: oneMoreSS(X, Y) :- lengthShape(Shape,X),
    lengthShape(Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- lengthShape(Shape,X), sizeShape(
    Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- sizeShape(Shape,X), heightShape(
    Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- sizeShape(Shape,X), widthShape(
    Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- sizeShape(Shape,X), lengthShape(
    Shape,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSS(X, Y) :- sizeShape(Shape,X), sizeShape(
    Shape,Y), minus(X, Y, Z), Z is 1.

precompute: oneMoreSP(X, Y) :- heightShape(Shape,X), height(
    Part,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- heightShape(Shape,X), width(Part
    ,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- heightShape(Shape,X), length(
    Part,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- heightShape(Shape,X), size(Part,
    Y), minus(X, Y, Z), Z is 1.

precompute: oneMoreSP(X, Y) :- widthShape(Shape,X), height(Part
    ,Y), minus(X, Y, Z), Z is 1.

```

```

precompute: oneMoreSP(X, Y) :- widthShape(Shape,X), width(Part,
    Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- widthShape(Shape,X), length(Part
    ,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- widthShape(Shape,X), size(Part,Y
    ), minus(X, Y, Z), Z is 1.

precompute: oneMoreSP(X, Y) :- lengthShape(Shape,X), height(
    Part,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- lengthShape(Shape,X), width(Part
    ,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- lengthShape(Shape,X), length(
    Part,Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- lengthShape(Shape,X), size(Part,
    Y), minus(X, Y, Z), Z is 1.

precompute: oneMoreSP(X, Y) :- sizeShape(Shape,X), height(Part,
    Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- sizeShape(Shape,X), width(Part,Y
    ), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- sizeShape(Shape,X), length(Part,
    Y), minus(X, Y, Z), Z is 1.
precompute: oneMoreSP(X, Y) :- sizeShape(Shape,X), size(Part,Y)
    , minus(X, Y, Z), Z is 1.

```

```

precompute: oneMorePS(Y, X) :- heightShape(Shape,X), height(
    Part,Y), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- heightShape(Shape,X), width(Part
    ,Y), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- heightShape(Shape,X), length(
    Part,Y), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- heightShape(Shape,X), size(Part,
    Y), minus(Y, X, Z), Z is 1.

precompute: oneMorePS(Y, X) :- widthShape(Shape,X), height(Part
    ,Y), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- widthShape(Shape,X), width(Part,
    Y), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- widthShape(Shape,X), length(Part
    ,Y), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- widthShape(Shape,X), size(Part,Y
    ), minus(Y, X, Z), Z is 1.

precompute: oneMorePS(Y, X) :- lengthShape(Shape,X), height(
    Part,Y), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- lengthShape(Shape,X), width(Part
    ,Y), minus(Y, X, Z), Z is 1.

```

```

precompute: oneMorePS(Y, X) :- lengthShape(Shape,X), length(
    Part,Y), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- lengthShape(Shape,X), size(Part,
    Y), minus(Y, X, Z), Z is 1.

precompute: oneMorePS(Y, X) :- sizeShape(Shape,X), height(Part,
    Y), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- sizeShape(Shape,X), width(Part,Y
    ), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- sizeShape(Shape,X), length(Part,
    Y), minus(Y, X, Z), Z is 1.
precompute: oneMorePS(Y, X) :- sizeShape(Shape,X), size(Part,Y)
    , minus(Y, X, Z), Z is 1.

```

### C.3 Planner

Below we present the list of predicates in the JSHOP2 planner.

```

(row ?x-loc ?y-loc ?z-loc ?width ?color)
(tower ?x-loc ?y-loc ?z-loc ?height ?color)
(column ?x-loc ?y-loc ?z-loc ?length ?color)
(square ?x-loc ?y-loc ?z-loc ?width ?color)
(rectangle ?x-loc ?y-loc ?z-loc ?width ?height ?color)
(cube ?x-loc ?y-loc ?z-loc ?width ?color)
(cuboid ?x-loc ?y-loc ?z-loc ?height ?width ?length ?color)
(block ?x-loc ?y-loc ?z-loc ?color)

```

## C.4 Concept Learning

Das et al. (2020) uses normalized compression distance (NCD) as a metric to compute concept similarity. Goldman and Kuter (2015) proposed NCD as a measure to compute diversity in the plans. NCD is computed as

$$\text{NCD}(x, y) = \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))}, \quad (\text{C.1})$$

where  $C(\cdot)$  is the length of the compressed file containing the string. NCD is able to capture the diversity of the plans. For example, consider the following plans.

$$\begin{aligned} p_1 &= (\text{drive } t1 \text{ a b}), (\text{drive } t1 \text{ b c}) \\ p_2 &= (\text{drive } t2 \text{ a b}), (\text{drive } t2 \text{ b c}) \\ p_3 &= (\text{fly } a1 \text{ a b}), (\text{drive, } t3, \text{ b, c}) \end{aligned}$$

NCD captures the expectation that the plan pair  $(p1, p3)$  is more diverse than the pair  $(p1, p2)$ , that is  $\text{NCD}(p1, p3) > \text{NCD}(p1, p2)$ . However, as the NCD metric relies on the compression measure, it depends on the characters used in the plan. It fails to capture the diversity when the operator names are changed. For example, NCD fails to capture  $\text{NCD}(c1, c3) > \text{NCD}(c1, c2)$  when the above plans are re-written as follows.

$$\begin{aligned} c1 &= (\text{action1 } 1 \text{ a b}), (\text{action1 } 1 \text{ b c}) \\ c2 &= (\text{action1 } 2 \text{ a b}), (\text{action1 } 2 \text{ b c}) \\ c3 &= (\text{action2 } 3 \text{ a b}), (\text{action1 } 2 \text{ b c}) \end{aligned}$$

To overcome this limitation, we implemented the following workarounds.

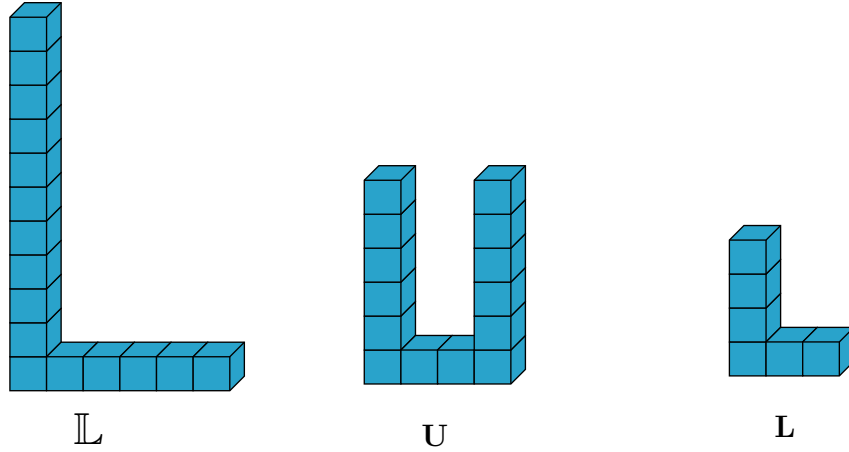


Figure C.1: Example of concepts.

1. **Initialized the concepts with the same parameters for comparisons.** Consider an  $\mathbb{L}$  shape of size  $10 \times 6$ , a  $\mathbf{U}$  shape of  $6 \times 3$  and an  $\mathbf{L}$  of size  $4 \times 3$ , shown in Figure C.1. As the number of blocks placed in the build plan of  $\mathbf{L}$  is closer to the number of blocks in  $\mathbf{U}$  than the  $\mathbb{L}$ , the  $\text{NCD}(\text{plan}(\mathbb{L}), \text{plan}(\mathbf{L})) > \text{NCD}(\text{plan}(\mathbf{U}), \text{plan}(\mathbf{L}))$ . To overcome this, we ensure the same sizes are used when two plans are compared.
2. **Actions in the plan had to be in same order.** NCD metric is sensitive to the order of actions in the plan. So an L shape constructed by placing a tower and then a row is considered different from an L shape constructed by placing a row and then a tower. To overcome this, we ensured the planner always plans to place blocks from bottom to top and left to right.
3. **Colors had to be removed before concept comparisons.** As the NCD measure is sensitive to the string, two identical plans with different colored blocks could be considered diverse. Hence, we removed the colors from the plan before comparison.
4. **Block numbers ranges were made to overlap.** As indicated above, the NCD measure was sensitive to the symbols used to represent the objects. Two identical

plans with block identifiers ranging from *b1–b10* were considered different from plans with block identifiers in the range *b22–b32*. So for faithful comparison, we used the same block identifier ranges when comparing plans.

In light of these workarounds, we do not recommend the NCD measure for concept learning. Devising an appropriate measure for concept learning is an important avenue of research.

## APPENDIX D

### ORIENT

#### D.1 Concave over modular mutual information

$f_\eta(A)$  is a restricted submodular function (Kothawade et al., 2022) defined over sets  $V, V'$  and  $\psi$  is a concave function. Let  $n$  be the size of set  $A$ .

$$\begin{aligned} f_\eta(A) = & \eta \sum_{i \in V'} \max \left( \psi \left( \sum_{j \in A \cap V} S_{ij} \right), \psi \left( \sqrt{n} \sum_{j \in A \cap V'} S_{ij} \right) \right) \\ & + \sum_{i \in V} \max \left( \psi \left( \sum_{j \in A \cap V'} S_{ij} \right), \psi \left( \sqrt{n} \sum_{j \in A \cap V} S_{ij} \right) \right) \end{aligned} \quad (\text{D.1})$$

#### D.2 Proof of the theorems

**Theorem.** *When the outer level loss of the discrete bi-level optimization problem of GLISTER is hinge loss, logistic loss, and perceptron loss, then the optimization problem becomes an instance of maximization of Concave over Modular (COM) SMI function.*

*Proof.* Given, training loss  $L_T$ , validation loss  $L_V$ , training set  $\mathcal{D}$ , subset  $\mathcal{S}$ , subset size  $k$ , and validation set  $\mathcal{V}$ , the objective of GLISTER can be written as follows:

$$\begin{aligned} & \min_{\mathcal{S} \subseteq \mathcal{D}, |\mathcal{D}|=k} L_V(\theta^*, \mathcal{V}) \\ & \text{where } \theta^* = \arg \min_{\theta} L_T(\theta, \mathcal{S}) \end{aligned} \quad (\text{D.2})$$

Loss on the subset denoted by  $L_T(\theta, \mathcal{S})$  is the summation of the losses of individual data samples in the subset. i.e.,  $L_T(\theta, \mathcal{S}) = \sum_{(x,y) \in \mathcal{S}} L_T(\theta, x, y)$ . Using the one-step gradient approximation of GLISTER, the above optimization problem can be written as:

$$\begin{aligned} & \min_{\mathcal{S} \subseteq \mathcal{D}, |\mathcal{D}|=k} L_V(\theta - \eta \nabla_{\theta} L_T(\theta, \mathcal{S}), \mathcal{V}) \\ & \min_{\mathcal{S} \subseteq \mathcal{D}, |\mathcal{D}|=k} L_V(\theta - \eta \sum_{(x,y) \in \mathcal{S}} \nabla_{\theta} L_T(\theta, x, y), \mathcal{V}) \end{aligned} \quad (\text{D.3})$$

where  $\eta$  is the learning rate.

We can convert the above minimization problem to a maximization problem as the following:

$$\max_{S \subseteq \mathcal{D}, |\mathcal{D}|=k} -L_V(\theta - \eta \sum_{(x,y) \in S} \nabla_{\theta} L_T(\theta, x, y), \mathcal{V}) \quad (\text{D.4})$$

Using the Proof of Theorem-1 in GLISTER (Killamsetty et al., 2021), the optimization problem in Equation D.4 for different validation losses can be written as follows:

**Case 1.** When  $L_V$  is hinge loss or perceptron loss, the optimization problem of GLISTER is:

$$\begin{aligned} \max_{S \subseteq \mathcal{D}, |\mathcal{D}|=k} \sum_{i=1}^{|\mathcal{V}|} \min(0, C_i + \sum_{j \in S} \hat{g}_{ij}) \\ \text{where } \sum_{j \in S} \hat{g}_{ij} \geq 0 \end{aligned} \quad (\text{D.5})$$

We can write the above optimization problem as,

$$\max_{S \subseteq \mathcal{D}, |\mathcal{D}|=k} |\mathcal{V}| C_i + \sum_{i=1}^{|\mathcal{V}|} \min(-C_i, \sum_{j \in S} \hat{g}_{ij}) \quad (\text{D.6})$$

$$\max_{S \subseteq \mathcal{D}, |\mathcal{D}|=k} \sum_{i=1}^{|\mathcal{V}|} \min(-C_i, \sum_{j \in S} \hat{g}_{ij}) \quad (\text{D.7})$$

Note that in the above equation,  $\min(C, x)$  is a concave function in  $x$  and  $\sum_{j \in S} \hat{g}_{ij}$  is a non-negative modular function. Hence, the above formulation is submodular and is an instance of concave over modular function.

**Case 2.** When  $L_V$  is logistic loss, the optimization problem of GLISTER is:

$$\max_{S \subseteq \mathcal{D}, |\mathcal{D}|=k} \sum_{i=1}^{|\mathcal{V}|} C - \log(1 + C_i \exp(\alpha \sum_{j \in S} \hat{g}_{ij})) \quad (\text{D.8})$$

Note that in the above equation,  $-\log(1 + C \exp -x)$  is concave in  $x$  and  $\sum_{j \in S} \hat{g}_{ij}$  is modular. Hence, the above formulation of set function is submodular and is an instance of concave over modular function. In our setting, we use labeled target data  $D^t$  as the validation set. Furthermore, the above given formulations of GLISTER corresponds to instantiation of maximization of COM MI function  $(\eta \sum_{i \in A} \psi(\sum_{j \in D^t} S_{ij}) + \sum_{j \in D^t} \psi(\sum_{i \in A} S_{ij}))$  with  $\eta = 0$ .

□

**Theorem.** *When the optimization problem of GRADMATCH with equal sample weights is used to match gradients of a held-out validation set, it becomes an instance of maximization of summation of GCMI and a diversity function.*

*Proof.* Given, training loss  $L_T$ , validation loss  $L_V$ , training set  $\mathcal{D}$ , and validation set  $\mathcal{V}$ . Let us denote loss of  $i^{th}$  sample in the training dataset as  $L_T^i(\theta)$  and the loss of  $j^{th}$  sample in the validation dataset as  $L_V^j(\theta)$ .

The objective of GRADMATCH with equal sample weights can be written as follows:

$$\min_{\mathcal{S} \subseteq \mathcal{D}, |\mathcal{S}|=k} \left\| \frac{1}{k} \sum_{i \in \mathcal{S}} \nabla_{\theta} L_T^i(\theta) - \frac{1}{|\mathcal{V}|} \sum_{j \in \mathcal{V}} \nabla_{\theta} L_V^j(\theta) \right\|^2 \quad (\text{D.9})$$

Without the loss of generality we assumed sample weights to be 1 in the above equation.

We can convert this to a maximization problem as follows:

$$\max_{\mathcal{S} \subseteq \mathcal{D}, |\mathcal{S}|=k} - \left\| \frac{1}{k} \sum_{i \in \mathcal{S}} \nabla_{\theta} L_T^i(\theta) - \frac{1}{|\mathcal{V}|} \sum_{j \in \mathcal{V}} \nabla_{\theta} L_V^j(\theta) \right\|^2 \quad (\text{D.10})$$

$$\max_{\mathcal{S} \subseteq \mathcal{D}, |\mathcal{S}|=k} \frac{-1}{k^2} \left\| \sum_{i \in \mathcal{S}} \nabla_{\theta} L_T^i(\theta) \right\|^2 + \frac{-1}{|\mathcal{V}|^2} \left\| \sum_{j \in \mathcal{V}} \nabla_{\theta} L_V^j(\theta) \right\|^2 + 2 \frac{1}{k|\mathcal{V}|} \sum_{i \in \mathcal{S}, j \in \mathcal{V}} \nabla_{\theta} L_T^i(\theta)^T \cdot \nabla_{\theta} L_V^j(\theta) \quad (\text{D.11})$$

In the above equation second term is not dependent on  $\mathcal{S}$  and can be ignored during optimization. Following which the above optimization problem can be written as:

$$\max_{\mathcal{S} \subseteq \mathcal{D}, |\mathcal{S}|=k} \frac{-1}{k^2} \left\| \sum_{i \in \mathcal{S}} \nabla_{\theta} L_T^i(\theta) \right\|^2 + 2 \frac{1}{k|\mathcal{V}|} \sum_{i \in \mathcal{S}, j \in \mathcal{V}} \nabla_{\theta} L_T^i(\theta)^T \cdot \nabla_{\theta} L_V^j(\theta) \quad (\text{D.12})$$

Note that in our setting, labeled target data  $D^t$  is used as the validation set. In the above equation, the second term corresponds to GCMI function( $2\lambda \sum_{i \in A, j \in D^t} S_{ij}$ ) with  $\lambda = 1$ .

Expanding on the first term we have,

$$\frac{-1}{k^2} \left\| \sum_{i \in \mathcal{S}} \nabla_{\theta} L_T^i(\theta) \right\|^2 = \frac{-1}{k^2} \sum_{i \in \mathcal{S}} \left\| \nabla_{\theta} L_T^i(\theta) \right\|^2 - \frac{2}{k^2} \sum_{i, j \in \mathcal{S} | i \neq j} \nabla_{\theta} L_T^i(\theta)^T \cdot \nabla_{\theta} L_T^j(\theta) \quad (\text{D.13})$$

The function given in Equation D.13 is a diversity function.

Hence, the optimization problem of GRADMATCH with equal sample weights when matched with validation set is a instance of maximization of summation of the GCMI function and diversity function.

□

**Theorem.** *When the optimization problem of CRAIG is adapted to match gradients of a held-out validation set, it becomes an instance of maximization of the FLMI function with  $\eta = 0$ .*

*Proof.* Given, training loss  $L_T$ , validation loss  $L_V$ , training set  $\mathcal{D}$ , and validation set  $\mathcal{V}$ . Let us denote loss of  $i^{th}$  sample in the training dataset as  $L_T^i(\theta)$  and the loss of  $j^{th}$  sample in the validation dataset as  $L_V^j(\theta)$ .

The objective of CRAIG matching gradients of a held-out validation set is as follows:

$$\max_{\mathcal{S} \subseteq \mathcal{D}, |\mathcal{S}|=k} \sum_{i \in \mathcal{V}} \max_{j \in \mathcal{S}} \nabla_{\theta} L_V^i(\theta)^T \cdot \nabla_{\theta} L_T^j(\theta) \quad (\text{D.14})$$

Note that in our setting, labeled target data  $D^t$  is used as the validation set. The set function in the above equation corresponds to FLMI function( $\sum_{i \in D^t} \max_{j \in A} S_{ij} + \eta \sum_{i \in A} \max_{j \in D^t} S_{ij}$ ) with  $\eta = 0$ .

Hence, the optimization problem of CRAIG adapted to match gradients of a held-out validation set is an instance of maximization of the FLMI function with  $\eta = 0$ . □

### D.3 SDA loss

**CCSA:** In CCSA, the classifier is modeled as a composition of two functions— $h \circ g$ . Here,  $g : X \rightarrow Z$  is a feature extractor that transforms the input from the feature space  $X$  to an embedding space  $Z$ , and  $h : Z \rightarrow Y$  is a predictor function. Let  $X_y^s$  and  $X_y^t$  denote the source

and the target domain samples with label  $y \in Y$ , respectively. The semantic alignment loss ( $\mathcal{L}_{SA}$ ) for CCSA is defined as,

$$\mathcal{L}_{SA}(g) = \sum_{y \in Y} d(P(g(X_y^s)), P(g(X_y^t))),$$

where  $d(\cdot)$  indicates a distance measure between the distributions of  $X_y^s$  and  $X_y^t$  in the embedding space and  $P(\cdot)$  indicates the distribution.  $\mathcal{L}_{SA}$  encourages the samples from different domains and the same label to map nearby in the embedding space. The separation loss ( $\mathcal{L}_S$ ) is defined as

$$\mathcal{L}_S(g) = \sum_{a, b \in Y | a \neq b} k(P(g(X_a^s)), P(g(X_b^t))),$$

where  $k$  is a similarity measure that returns a higher value when the distribution of  $X_a^s$  and  $X_b^t$  is close in the embedding space. Hence,  $\mathcal{L}_S$  encourages the samples from different domains and different labels to map far away in the embedding space. Overall the CCSA loss is defined as a combination of cross-entropy loss, semantic alignment loss, and separation loss,

$$\mathcal{L}_{CCSA}(h \circ g) = \mathcal{L}_{CE}(h \circ g) + \mathcal{L}_{SA}(g) + \mathcal{L}_S(g).$$

The distance measure ( $d$ ) in the semantic alignment loss and the similarity measure ( $k$ ) in the separation loss are computed as average of pairwise similarities and distances between all the samples from the source and the target domain, respectively. Further assumptions on  $d$  and  $k$  lends them into a well known contrastive loss function (cf. Motiian et al. (2017) for details).

**$d$ -SNE:** Instead of minimizing the average of distances between all pairs of samples, Xu et al. (2019), in  $d$ -SNE, proposes to minimize the largest distance of the samples from

different source with same label, and maximizing the smallest distance of the samples from different source and different labels. Let  $D_y^s$  denote the subset of source data with label  $y$ , then the loss function  $\tilde{L}(g)$  is defined as,

$$\tilde{L}(g) = \sum_{x_j \in D^t} \left( \sup_{x \in D_{y_j}^s} \{a \mid a \in d(g(x), g(x_j))\} - \inf_{x \in D^s \setminus D_{y_j}^s} \{b \mid b \in d(g(x), g(x_j))\} \right).$$

The complete  $d$ -SNE loss is defined as a combination of  $\tilde{L}$  and cross-entropy loss on source and target domain.

$$\mathcal{L}_{d\text{-SNE}}(h \circ g) = \tilde{\mathcal{L}}(g) + \alpha \mathcal{L}_{\text{CE}}^s(h \circ g) + \beta \mathcal{L}_{\text{CE}}^t(h \circ g).$$

#### D.4 Experiment details

We use a common training process and hyperparameters for all our experiments. We use the following hyperparameters:

- **Optimization Algorithm:** SGD with Momentum
- **Learning Rate:** 0.001 with Cosine Annealing
- **Momentum:** 0.9
- **Weight Decay:** 5e-4
- **SMI query diversity hyperparameter( $\eta$ ):** 1
- **Number of epochs:** 300

We use a single GTX 1080 Ti GPU for experiments.

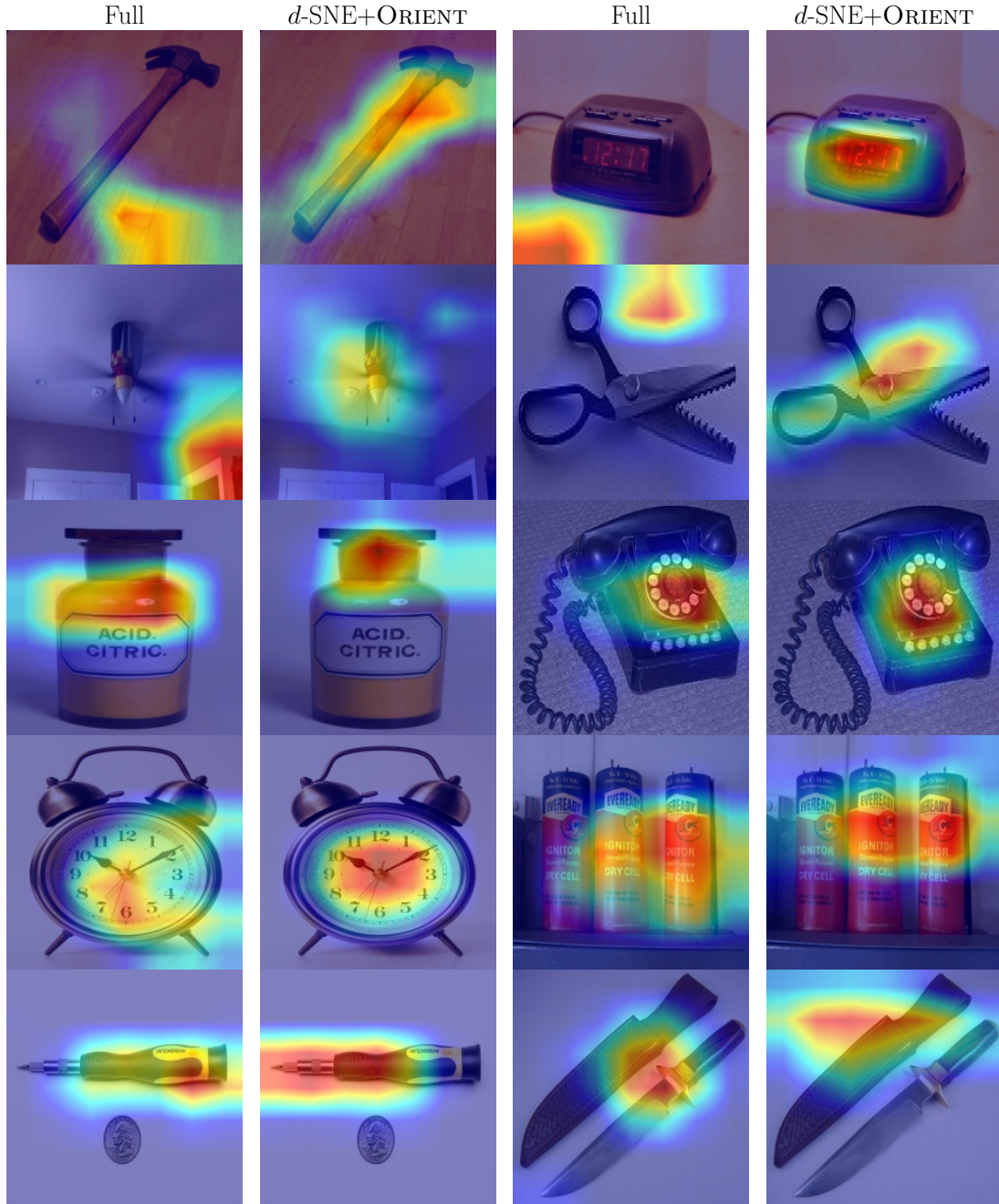


Figure D.1: GradCam (Selvaraju et al., 2017) activation maps of the models learned using  $d$ -SNE + Full and  $d$ -SNE + ORIENT (FLMI) on the Office-Home dataset with “Product” as the source domain and “Real World” as the target domain. As evidenced by class activation maps, the ORIENT framework enabled the model to learn more effective class discriminative features than Full data training consistently.

Table D.1: Test accuracy on Office-31 dataset.

	A $\rightarrow$ D	A $\rightarrow$ W	D $\rightarrow$ A	D $\rightarrow$ W	W $\rightarrow$ A	W $\rightarrow$ D
Full	$0.77 \pm 0.02$	$0.71 \pm 0.02$	$0.62 \pm 0.01$	$0.95 \pm 0.01$	$0.61 \pm 0.01$	$0.98 \pm 0.01$
Random	$0.63 \pm 0.03$	$0.57 \pm 0.04$	$0.54 \pm 0.02$	$0.86 \pm 0.02$	$0.5 \pm 0.03$	$0.91 \pm 0.02$
ORIENT (FLMI)	$0.79 \pm 0.02$	$0.73 \pm 0.01$	$0.62 \pm 0.01$	$0.93 \pm 0.01$	$0.59 \pm 0$	$0.96 \pm 0.01$
ORIENT (GC)	$0.52 \pm 0.12$	$0.5 \pm 0.08$	$0.39 \pm 0.07$	$0.69 \pm 0.11$	$0.42 \pm 0.09$	$0.72 \pm 0.08$
ORIENT (L)	$0.74 \pm 0.03$	$0.72 \pm 0.02$	$0.63 \pm 0.01$	$0.93 \pm 0.01$	$0.58 \pm 0.01$	$0.97 \pm 0.01$
ORIENT (G)	$0.75 \pm 0.01$	$0.72 \pm 0.01$	$0.61 \pm 0.01$	$0.92 \pm 0.01$	$0.58 \pm 0.01$	$0.96 \pm 0$
ORIENT (GM)	$0.77 \pm 0.02$	$0.73 \pm 0.02$	$0.61 \pm 0.01$	$0.93 \pm 0.01$	$0.61 \pm 0.01$	$0.97 \pm 0.01$

Table D.2: Training time (in hours) for 300 epochs on Office-31 dataset

	A $\rightarrow$ D	A $\rightarrow$ W	D $\rightarrow$ A	D $\rightarrow$ W	W $\rightarrow$ A	W $\rightarrow$ D
Full	$3.13 \pm 0.02$	$3.04 \pm 0.02$	$1.47 \pm 0.52$	$1.37 \pm 0$	$1.81 \pm 1.12$	$1.25 \pm 0.01$
Random	$0.95 \pm 0.01$	$0.92 \pm 0$	$0.77 \pm 0.04$	$0.48 \pm 0.13$	$0.73 \pm 0.01$	$0.39 \pm 0$
ORIENT (FLMI)	$1.04 \pm 0.09$	$0.82 \pm 0.18$	$0.89 \pm 0.01$	$0.46 \pm 0.01$	$0.34 \pm 0.01$	$0.43 \pm 0$
ORIENT (GC)	$1.02 \pm 0.02$	$0.97 \pm 0$	$0.84 \pm 0.01$	$0.46 \pm 0$	$0.7 \pm 0.02$	$0.43 \pm 0.01$
ORIENT (L)	$1.53 \pm 0$	$1.48 \pm 0.02$	$0.81 \pm 0.01$	$0.46 \pm 0$	$0.71 \pm 0.01$	$0.45 \pm 0$
ORIENT (G)	$1.04 \pm 0.02$	$0.99 \pm 0$	$0.55 \pm 0.11$	$0.47 \pm 0$	$0.35 \pm 0.01$	$0.43 \pm 0$
ORIENT (GM)	$1.16 \pm 0.03$	$1.08 \pm 0.01$	$0.86 \pm 0.22$	$0.56 \pm 0.02$	$0.35 \pm 0$	$0.49 \pm 0.01$

## D.5 Additional results

Figure D.1 presents the GradCam (Selvaraju et al., 2017) class-activation maps of trained models on the Office-Home dataset ( $P \rightarrow R$  setting) using  $d$ -SNE loss for both Full and ORIENT (FLMI). These activation maps show that the model trained with ORIENT framework learn effective class discriminative features more consistently than Full.

We present the test accuracy of models trained using cross-entropy loss on a combination of source and target domain data,  $D^s \cup D^t$ , of Office-31 and Office-Home datasets in D.1 and D.5, respectively. We performed 3 runs of each experiment using different initial training data subset each time. We see that ORIENT(FLMI) and ORIENT(GM) perform similar to Full and outperform Random across all experiments. Tables D.2 and D.6 show the training times for these settings. We see that all instantiations of ORIENT except ORIENT(L) achieve  $\sim 3\times$  speed-up compared to Full.

Table D.3: Test accuracy on Office-31 dataset with SDA methods

		A $\rightarrow$ D	A $\rightarrow$ W	D $\rightarrow$ A	D $\rightarrow$ W	W $\rightarrow$ A	W $\rightarrow$ D
CCSA	Full	$0.78 \pm 0$	$0.72 \pm 0$	$0.55 \pm 0$	$0.93 \pm 0$	$0.55 \pm 0$	$0.97 \pm 0$
	Random	$0.76 \pm 0.01$	$0.72 \pm 0.01$	$0.54 \pm 0.02$	$0.81 \pm 0.02$	$0.54 \pm 0.01$	$0.92 \pm 0.02$
	ORIENT (G)	$0.78 \pm 0.03$	$0.73 \pm 0.02$	$0.55 \pm 0.04$	$0.92 \pm 0.01$	$0.56 \pm 0$	$0.97 \pm 0.01$
	ORIENT (GM)	$0.78 \pm 0.01$	$0.74 \pm$	$0.51 \pm 0.04$	$0.88 \pm 0.02$	$0.55 \pm 0.02$	$0.97 \pm 0.02$
	ORIENT (FLMI)	$0.77 \pm 0$	$0.76 \pm 0.01$	$0.55 \pm 0$	$0.89 \pm 0.02$	$0.55 \pm 0.02$	$0.96 \pm 0.01$
$d$ -SNE	Full	$0.77 \pm 0$	$0.69 \pm 0$	$0.53 \pm 0.01$	<b><math>0.93 \pm 0</math></b>	$0.54 \pm 0$	$0.98 \pm 0$
	Random	$0.76 \pm 0.02$	$0.68 \pm 0.02$	$0.53 \pm 0.02$	$0.86 \pm 0.02$	$0.53 \pm 0.01$	$0.94 \pm 0.02$
	ORIENT (G)	$0.73 \pm 0$	$0.69 \pm 0.01$	$0.52 \pm 0.02$	<b><math>0.93 \pm 0.01</math></b>	$0.54 \pm 0$	$0.97 \pm 0.01$
	ORIENT (GM)	$0.76 \pm 0.01$	$0.69 \pm$	$0.52 \pm 0.04$	$0.89 \pm 0.02$	$0.54 \pm 0.02$	$0.97 \pm 0.01$
	ORIENT (FLMI)	$0.78 \pm 0$	$0.71 \pm 0.01$	<b><math>0.55 \pm 0</math></b>	$0.90 \pm 0.02$	$0.56 \pm 0.02$	$0.97 \pm 0.01$

Table D.4: Training time in hours on Office-31 with SDA methods

		A $\rightarrow$ D	A $\rightarrow$ W	D $\rightarrow$ A	D $\rightarrow$ W	W $\rightarrow$ A	W $\rightarrow$ D
CCSA	Full	$24.11 \pm 0.02$	$27.81 \pm 1.0$	$7.16 \pm 0.1$	$7.84 \pm 0.01$	$10.46 \pm 0.01$	$10.54 \pm 1.15$
	Random	$14.71 \pm 2.5$	$14.64 \pm 0.19$	$3.37 \pm 0.02$	$2.64 \pm 0.39$	$4.11 \pm 0.08$	$3.25 \pm 0.77$
	ORIENT (G)	$16.59 \pm 0.42$	$15.53 \pm 1.2$	$3.63 \pm 0$	$3.45 \pm 0.48$	$4.53 \pm 0.21$	$4.14 \pm 0.77$
	ORIENT (GM)	$17.11 \pm 1.11$	$15.6 \pm$	$3.14 \pm 0.95$	$4.1 \pm 0.34$	$5.33 \pm 0.6$	$5.21 \pm 1.29$
	ORIENT (FLMI)	$13.93 \pm 8.85$	$13.24 \pm 8.47$	$3.79 \pm 0.65$	$3.53 \pm 1.3$	$5.34 \pm 1.27$	$5.21 \pm 1.25$
$d$ -SNE	Full	$11.57 \pm 0.02$	$8.33 \pm 0.5$	$2.01 \pm 0.1$	$2.21 \pm 0.1$	$2.18 \pm 0.1$	$3.42 \pm 0.2$
	Random	$3.49 \pm 0.05$	$2.48 \pm 0.01$	$0.61 \pm 0.02$	$0.65 \pm 0.013$	$0.69 \pm 0.01$	$1.04 \pm 0.02$
	ORIENT (G)	$3.53 \pm 0.05$	$2.5072 \pm 0.02$	$0.66 \pm 0.05$	$0.69 \pm 0.01$	$0.72 \pm 0.01$	$1.06 \pm 0.01$
	ORIENT (GM)	$3.56 \pm 0.01$	$2.56 \pm 0.01$	$0.68 \pm 0.02$	$0.71 \pm 0.03$	$0.73 \pm 0.03$	$1.10 \pm 0.02$
	ORIENT (FLMI)	$3.55 \pm 0.01$	$2.53 \pm 0.04$	$0.66 \pm 0.01$	$0.69 \pm 0.01$	$0.72 \pm 0.02$	$1.06 \pm 0.02$

Table D.5: Test accuracy on Office-Home dataset

	R $\rightarrow$ P	R $\rightarrow$ C	P $\rightarrow$ R	P $\rightarrow$ C	C $\rightarrow$ R	C $\rightarrow$ P	A $\rightarrow$ P	A $\rightarrow$ R	A $\rightarrow$ C	R $\rightarrow$ A	P $\rightarrow$ A	C $\rightarrow$ A
Full	0.79	0.5	0.62	<b>0.49</b>	0.59	0.68	0.71	0.63	<b>0.5</b>	<b>0.58</b>	0.48	0.43
Random	0.69	0.39	0.58	0.37	0.5	0.58	0.61	0.55	0.38	0.46	0.42	0.34
ORIENT (G)	0.77	0.5	<b>0.66</b>	0.48	0.61	0.67	0.71	0.63	0.49	0.55	<b>0.53</b>	0.42
ORIENT (GM)	0.76	0.5	0.64	0.47	0.59	0.71	<b>0.73</b>	<b>0.67</b>	0.48	0.57	0.5	<b>0.46</b>
ORIENT (FLMI)	0.79	<b>0.52</b>	0.63	0.48	<b>0.62</b>	<b>0.69</b>	0.72	0.65	<b>0.5</b>	0.55	0.45	<b>0.46</b>

Table D.6: Training time(in hours) for 300 epochs on Office-Home dataset

	R $\rightarrow$ P	R $\rightarrow$ C	P $\rightarrow$ R	P $\rightarrow$ C	C $\rightarrow$ R	C $\rightarrow$ P	A $\rightarrow$ P	A $\rightarrow$ R	A $\rightarrow$ C	R $\rightarrow$ A	P $\rightarrow$ A	C $\rightarrow$ A
Full	44.28	61.58	37.72	57.2	18.47	59.6	14.86	13.96	9.61	27.55	23.97	20.32
Random	6.92	21.34	13.8	11.72	7.16	17.5	7.6	7.11	5.79	10.32	14.3	10.27
ORIENT (G)	14.41	13.65	15.57	9.95	6.93	12.61	8.46	8.29	6.8	11.49	16.28	11.42
ORIENT (GM)	15.88	26.15	17	22.78	10.02	13.18	9.77	9.23	7.38	12.48	17.34	12.02
ORIENT (FLMI)	7.78	25.84	15.73	13.33	7.02	19.97	8.85	8.45	6.62	11.47	16.33	11.31

Table D.7: Test accuracy on Office-Home with SDA methods

		R $\rightarrow$ P	R $\rightarrow$ C	P $\rightarrow$ R	P $\rightarrow$ C	C $\rightarrow$ R	C $\rightarrow$ P	A $\rightarrow$ P	A $\rightarrow$ R	A $\rightarrow$ C	R $\rightarrow$ A	P $\rightarrow$ A	C $\rightarrow$ A
CCSA	Full	0.74	<b>0.55</b>	0.62	0.46	0.56	0.67	0.70	0.64	0.48	0.57	0.49	0.41
	Random	0.71	0.47	0.62	0.45	0.54	0.66	0.69	0.57	0.48	0.5	0.47	0.45
	ORIENT(G)	<b>0.78</b>	0.54	0.63	<b>0.5</b>	0.59	0.7	<b>0.72</b>	0.62	0.5	0.57	0.49	<b>0.5</b>
	ORIENT(GM)	0.75	0.5	0.63	0.48	0.59	0.69	0.69	0.63	0.49	0.51	0.5	0.46
	ORIENT(FLMI)	<b>0.78</b>	<b>0.54</b>	<b>0.65</b>	<b>0.5</b>	<b>0.61</b>	<b>0.71</b>	0.71	<b>0.65</b>	<b>0.51</b>	<b>0.59</b>	<b>0.54</b>	0.47
$d$ -SNE	Full	<b>0.77</b>	<b>0.53</b>	<b>0.62</b>	<b>0.50</b>	<b>0.60</b>	<b>0.71</b>	<b>0.72</b>	<b>0.63</b>	0.49	0.52	0.44	0.40
	Random	0.75	0.50	0.60	0.45	0.57	0.69	0.68	0.59	0.49	0.46	0.43	0.40
	ORIENT(G)	0.75	0.51	0.62	0.49	0.59	0.7	<b>0.72</b>	0.62	0.5	<b>0.54</b>	0.44	0.41
	ORIENT(GM)	0.76	<b>0.52</b>	0.62	<b>0.50</b>	0.59	0.70	0.71	0.61	0.49	0.51	<b>0.46</b>	<b>0.42</b>
	ORIENT(FLMI)	<b>0.77</b>	<b>0.52</b>	<b>0.63</b>	<b>0.50</b>	<b>0.60</b>	<b>0.71</b>	<b>0.71</b>	0.61	<b>0.51</b>	0.52	0.44	<b>0.42</b>

Table D.8: Training time(in hours) on Office-Home with SDA methods

		R $\rightarrow$ P	R $\rightarrow$ C	P $\rightarrow$ R	P $\rightarrow$ C	C $\rightarrow$ R	C $\rightarrow$ P	A $\rightarrow$ P	A $\rightarrow$ R	A $\rightarrow$ C	R $\rightarrow$ A	P $\rightarrow$ A	C $\rightarrow$ A
CCSA	Full	17.73	17.53	18.23	14.93	17.25	16.42	10.32	9.45	16.13	17.67	8.29	11.35
	Random	5.57	5.17	4.74	4.87	5.57	4.62	3.48	3.12	4.25	5.88	2.52	3.45
	ORIENT (G)	6.01	6.94	5.78	5.15	6.61	5.92	3.71	3.41	4.61	6.13	2.87	4.12
	ORIENT (GM)	5.73	5.69	5.15	5.39	6.12	5.34	3.78	3.85	4.66	6.27	3.17	3.95
	ORIENT (FLMI)	6.12	6.13	5.73	6.17	5.56	5.99	3.94	3.34	4.89	6.72	2.78	4.23
$d$ -SNE	Full	16.84	16.68	17.42	9.55	17.05	9.60	6.64	9.94	6.29	18.81	11.69	11.27
	Random	5.23	5.31	5.26	2.87	5.12	2.88	1.99	3.11	1.90	5.85	3.55	3.41
	ORIENT (G)	5.48	5.61	5.42	2.98	5.32	2.96	2.11	3.48	1.95	6.21	3.63	3.53
	ORIENT (GM)	6.08	6.01	5.50	3.09	5.51	3.12	2.17	3.65	2.08	6.62	3.81	3.93
	ORIENT (FLMI)	5.56	5.48	5.44	3.01	5.34	2.98	2.07	3.44	2.04	6.23	3.67	3.58

We present test accuracy of models trained using SDA loss on source domain data,  $D^s$ , of Office-31 dataset in D.3. Again, we see that all instantiations of ORIENT perform similar

to Full while outperforming Random. Table D.4 shows the training times for this setting. Again, we see that all instantiations of ORIENT achieve  $\sim 2.5\times$  speed-up compared to Full.

Finally, Table D.7 presents the test accuracy of models trained using SDA loss on source domain data,  $D^s$ , of Office-Home dataset. Here, we see that instantiations of ORIENT perform substantially better than Full and Random in some experiments and perform similar to Full in the rest. In particular, ORIENT(FLMI) outperforms Full in the settings of  $R \rightarrow P$ ,  $C \rightarrow R$ ,  $C \rightarrow P$  and  $P \rightarrow A$ . Table D.8 presents the corresponding training times. Once again, we see that instantiations of ORIENT achieve  $\sim 3\times$  speed-up compared to Full.

### D.5.1 Synthetic experiments

To provide better intuition into how different SMI functions select data subsets, we present a comparison of the subset selected for two toy datasets in Figures D.2 and D.3. Figure D.2a and D.3a presents the synthetic dataset with 5000 samples in the source domain. The target domain consists of the 500 data points highlighted with a different color. The query set is of size 50. We present the subset selected by ORIENT with Facility Location Mutual Information (**ORIENT (FLMI)**), GradMatch (**ORIENT (GM)**), GLISTER (**ORIENT (G)**), Log Determinant Mutual Information (**ORIENT (LDMI)**), and Graph Cut Mutual Information (**ORIENT (GCMI)**). From the results, it is evident that the FLMI function selects sample sources closer to the target domain than other SMI functions. In contrast, the GM function selects representative samples from the source domain. Our results show that the G function selects samples near the decision boundaries of the source domain data. The GCMI function selects samples from a source domain that are very similar and clustered together. In synthetic results, we found that the LDMI function tends to prioritize the selection of data samples from certain classes compared to others.

### D.5.2 Analysis of data subset size

Table D.9 presents the target domain accuracies achieved using different subset sizes on the office-31 dataset using d-SNE loss, and Table D.10 presents the training times taken by using different subset sizes on the office-31 dataset using d-SNE loss. As seen, on reducing the subset size from 0.3 fraction to 0.1, model experiences loss of accuracy but gains in terms of training time. Hence, there is a trade-off between training time and accuracy of the model. Higher fraction of the data would lead to better performance in accuracy but also require more training time. Where as, lower fraction would require less training time but might result in lower accuracy.

Table D.9: Comparison of test accuracy for office-31 with d-SNE loss function and different fractions of subset selection.

	Fraction	A $\rightarrow$ D	A $\rightarrow$ W	D $\rightarrow$ A	D $\rightarrow$ W	W $\rightarrow$ A	W $\rightarrow$ D
Full	1.0	0.77	0.69	0.53	0.93	0.54	0.98
Random	0.1	0.65	0.58	0.43	0.62	0.48	0.72
Random	0.3	0.76	0.68	0.53	0.86	0.53	0.94
ORIENT (FLMI)	0.1	0.70	0.67	0.50	0.77	0.48	0.92
ORIENT (FLMI)	0.3	0.78	0.71	0.55	0.90	0.56	0.97
ORIENT (G)	0.1	0.75	0.60	0.42	0.70	0.48	0.89
ORIENT (G)	0.3	0.76	0.66	0.50	0.87	0.52	0.96

Table D.10: Training time in hours on Office-31 with d-SNE loss function and different fractions of subset selection.

	Fraction	A $\rightarrow$ D	A $\rightarrow$ W	D $\rightarrow$ A	D $\rightarrow$ W	W $\rightarrow$ A	W $\rightarrow$ D
Full	1.0	11.57	8.33	2.01	2.21	2.18	3.42
Random	0.1	1.18	0.83	0.20	0.22	0.23	0.34
Random	0.3	3.49	2.48	0.61	0.65	0.69	1.04
ORIENT (FLMI)	0.1	1.25	0.89	0.25	0.26	0.26	0.38
ORIENT (FLMI)	0.3	3.55	2.53	0.66	0.69	0.72	1.06
ORIENT (G)	0.1	1.21	0.87	0.25	0.26	0.25	0.37
ORIENT (G)	0.3	3.53	2.51	0.66	0.69	0.72	1.06

### D.5.3 Analysis of $L$ for subset selection

We present comparison of target domain accuracy achieved by ORIENT (FLMI) for different  $L$  values of 5, 10, 20, 40 on Office 31 ( $A \rightarrow D$ ) using d-SNE loss and 30% subset in Table D.11. Note that smaller the value of  $L$  is, greater the frequency of subset selection. Results demonstrate that using  $L = 5, 10$  (i.e., more frequent subset selection) results in higher training time with no improvement in accuracy. Whereas using  $L = 40$  (i.e., less frequent subset selection) results in lower target domain accuracy with not much significant improvement in training time. Hence, we used  $L = 20$  in our experiments.

Table D.11: Table showing target domain accuracy and training time taken(in hrs) achieved on office-31 ( $A \rightarrow D$ ) using d-SNE loss function and 30% subset.

Method	Epoch Interval ( $L$ )	Target domain accuracy	Time Taken(in hrs)
ORIENT(FLMI)	5	0.78	3.75
ORIENT(FLMI)	10	0.78	3.61
ORIENT(FLMI)	20	0.78	3.55
ORIENT(FLMI)	40	0.77	3.51

### D.5.4 Analysis of time taken

Additionally, we present the convergence curve of training time against the validation loss in Fig. D.4. As different methods use different losses, the absolute values of loss are not directly comparable. But we still like to present these plots to show that even though the Full starts with reasonable performance in terms of accuracy (in Fig. D.4), it does not start with lowest validation loss, and multiple training epochs are necessary before it converges. The plots show that the training time required by the ORIENT methods to converge is consistently lower than the training time required by Full to converge.

Tables D.12, D.13, D.14, D.15, D.16, D.17, D.18, D.19, D.20, D.21, D.22, and D.23 present ratio of time taken by subset selection methods with respect to Full training to achieve test accuracy in the range of (0.3, 0.8) with increments of 0.05. We could see that

Table D.12: Setting:  $R \rightarrow P$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8
Random	6.31	6.31	6.31	6.31	6.31	6.31	6.31	3.13	0.51	0.0	0.0
CRAIG	4.78	4.78	4.78	4.78	4.78	4.78	0.27	0.0	0.0	0.0	0.0
ORIENT (FLMI)	6.17	6.17	6.17	6.17	6.17	6.17	6.17	6.17	1.92	6.21	3.21
ORIENT (G)	2.49	2.49	2.49	2.49	2.49	2.49	2.49	2.49	0.22	2.01	0.0
ORIENT (GM)	3.93	3.93	3.93	3.93	3.93	3.93	3.93	1.29	0.43	3.53	0.0

Table D.13: Setting:  $R \rightarrow C$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5
Random	3.22	3.22	1.62	0.0	0.0
CRAIG	0.58	0.0	0.0	0.0	0.0
ORIENT (FLMI)	3.18	3.18	1.48	1.48	0.35
ORIENT (G)	7.58	7.58	7.58	0.72	0.45
ORIENT (GM)	4.83	4.83	2.04	0.97	0.3

Table D.14: Setting:  $R \rightarrow A$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5	0.55
Random	3.38	3.38	3.38	3.38	3.38	0.0
CRAIG	4.7	4.7	4.7	4.7	0.0	0.0
ORIENT (FLMI)	3.42	3.42	3.42	3.42	0.75	0.42
ORIENT (G)	3.4	3.4	3.4	3.4	0.31	0.26
ORIENT (GM)	3.35	3.35	3.35	3.35	3.35	0.9

Speedups achieved by different subset selection strategies w.r.t Full training to reach different accuracy thresholds for different combinations using  $R$  as source domain on the Officehome dataset in the augmented setting.

ORIENT(FLMI) achieves a performance threshold faster than Full in 70 out of 82 cases, ORIENT(GM) achieves a performance threshold faster than Full in 65 out of 82 cases, ORIENT(G) achieves a performance threshold faster than Full in 60 out of 82 cases, whereas CRAIG achieves a performance threshold faster than Full in 27 out of 82 cases and Random achieves a performance threshold faster than Full in 56 out of 82 cases consisting only of lower accuracy thresholds. Furthermore, Random always fails to achieve similar accuracy to

Table D.15: Setting:  $P \rightarrow R$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65
Random	3.31	3.31	3.31	3.31	3.31	3.31	0.37	0.0
CRAIG	6.24	6.24	6.24	6.24	0.39	0.0	0.0	0.0
ORIENT (FLMI)	3.38	3.38	3.38	3.38	3.38	3.38	0.48	1.62
ORIENT (G)	3.25	3.25	3.25	3.25	3.25	3.25	0.32	1.54
ORIENT (GM)	3.39	3.39	3.39	3.39	3.39	3.39	1.38	1.72

Table D.16: Setting:  $P \rightarrow C$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5
Random	9.46	9.46	4.47	0.0	0.0
CRAIG	0.0	0.0	0.0	0.0	0.0
ORIENT (FLMI)	9.72	4.02	4.02	1.48	Improved
ORIENT (G)	8.88	8.88	1.27	0.91	0.0
ORIENT (GM)	4.02	4.02	4.02	0.47	0.0

Table D.17: Setting:  $P \rightarrow A$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5
Random	3.31	3.31	3.31	3.31	0.0
CRAIG	8.32	8.32	0.0	0.0	0.0
ORIENT (FLMI)	3.34	3.34	1.54	0.74	Improved
ORIENT (G)	3.3	3.3	3.3	0.36	Improved
ORIENT (GM)	3.28	3.28	3.28	1.39	Improved

Speedups achieved by different subset selection strategies w.r.t Full training to reach different accuracy thresholds for different combinations using  $P$  as source domain on the Officehome dataset in the augmented setting.

Full. From this, it is evident that ORIENT achieves faster performance thresholds than FULL in most cases.

Table D.24 presents the ratio of time taken by subset selection methods with respect to Full training to achieve validation loss within 105% of the minimum validation loss. Even with this impractical stopping criterion we see average speedups of 2.26, 2.37 and 1.85 for ORIENT(FLMI), ORIENT(G) and ORIENT(GM), respectively. Similarly, speedups

Table D.18: Setting:  $C \rightarrow R$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5	0.55	0.6
Random	3.38	3.38	3.38	3.38	1.69	0.0	0.0
CRAIG	2.26	2.26	0.17	0.0	0.0	0.0	0.0
ORIENT (FLMI)	3.33	3.33	3.33	3.33	1.49	1.49	2.99
ORIENT (G)	3.29	3.29	3.29	3.29	3.29	0.24	1.81
ORIENT (GM)	3.29	3.29	3.29	3.29	3.29	0.5	1.54

Table D.19: Setting:  $C \rightarrow P$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7
Random	2.22	2.22	2.22	2.22	2.22	0.44	0.0	0.0	0.0
CRAIG	9.98	9.98	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ORIENT (FLMI)	2.46	2.46	2.46	2.46	2.46	1.1	0.7	0.29	Improved
ORIENT (G)	3.09	3.09	3.09	3.09	3.09	0.37	0.3	0.24	0.0
ORIENT (GM)	3.2	3.2	3.2	3.2	3.2	1.34	0.69	0.69	Improved

Table D.20: Setting:  $C \rightarrow A$ 

Accuracy threshold	0.3	0.35	0.4	0.45
Random	3.34	1.66	0.0	0.0
CRAIG	0.0	0.0	0.0	0.0
ORIENT (FLMI)	3.38	3.38	0.75	Improved
ORIENT (G)	3.33	1.52	0.28	Improved
ORIENT (GM)	3.34	3.34	0.89	Improved

Speedups achieved by different subset selection strategies w.r.t Full training to reach different accuracy thresholds for different combinations using  $C$  as source domain on the Officehome dataset in the augmented setting.

achieved in reaching  $1.1\times$  minimum validation loss for ORIENT(FLMI), ORIENT(G) and ORIENT(GM) are 2.28, 2.38 and 1.85, respectively.

Table D.21: Setting:  $A \rightarrow R$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65
Random	3.28	3.28	3.28	3.28	3.28	1.64	0.0	0.0
CRAIG	5.26	5.26	5.26	5.26	0.45	0.29	0.0	0.0
ORIENT (FLMI)	3.34	3.34	3.34	3.34	3.34	3.34	1.48	0.0
ORIENT (G)	3.34	3.34	3.34	3.34	3.34	3.34	0.28	Improved
ORIENT (GM)	3.29	3.29	3.29	3.29	3.29	3.29	1.34	Improved

Table D.22: Setting:  $A \rightarrow P$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7
Random	3.77	3.77	3.77	3.77	3.77	3.77	1.8	0.0	0.0
CRAIG	15.32	15.32	2.53	0.32	0.0	0.0	0.0	0.0	0.0
ORIENT (FLMI)	3.77	3.77	3.77	3.77	3.77	3.77	1.6	0.98	0.33
ORIENT (G)	3.76	3.76	3.76	3.76	3.76	3.76	0.37	0.37	0.24
ORIENT (GM)	3.74	3.74	3.74	3.74	3.74	3.74	1.41	0.5	0.36

Table D.23: Setting:  $A \rightarrow C$ 

Accuracy threshold	0.3	0.35	0.4	0.45	0.5
Random	3.4	1.67	0.0	0.0	0.0
CRAIG	0.0	0.0	0.0	0.0	0.0
ORIENT (FLMI)	3.39	1.52	1.52	0.5	0.22
ORIENT (G)	3.36	3.36	0.36	0.36	0.0
ORIENT (GM)	3.32	3.32	1.35	0.41	0.19

Speedups achieved by different subset selection strategies w.r.t Full training to reach different accuracy thresholds for different combinations using  $A$  as source domain on the Officehome dataset in the augmented setting.

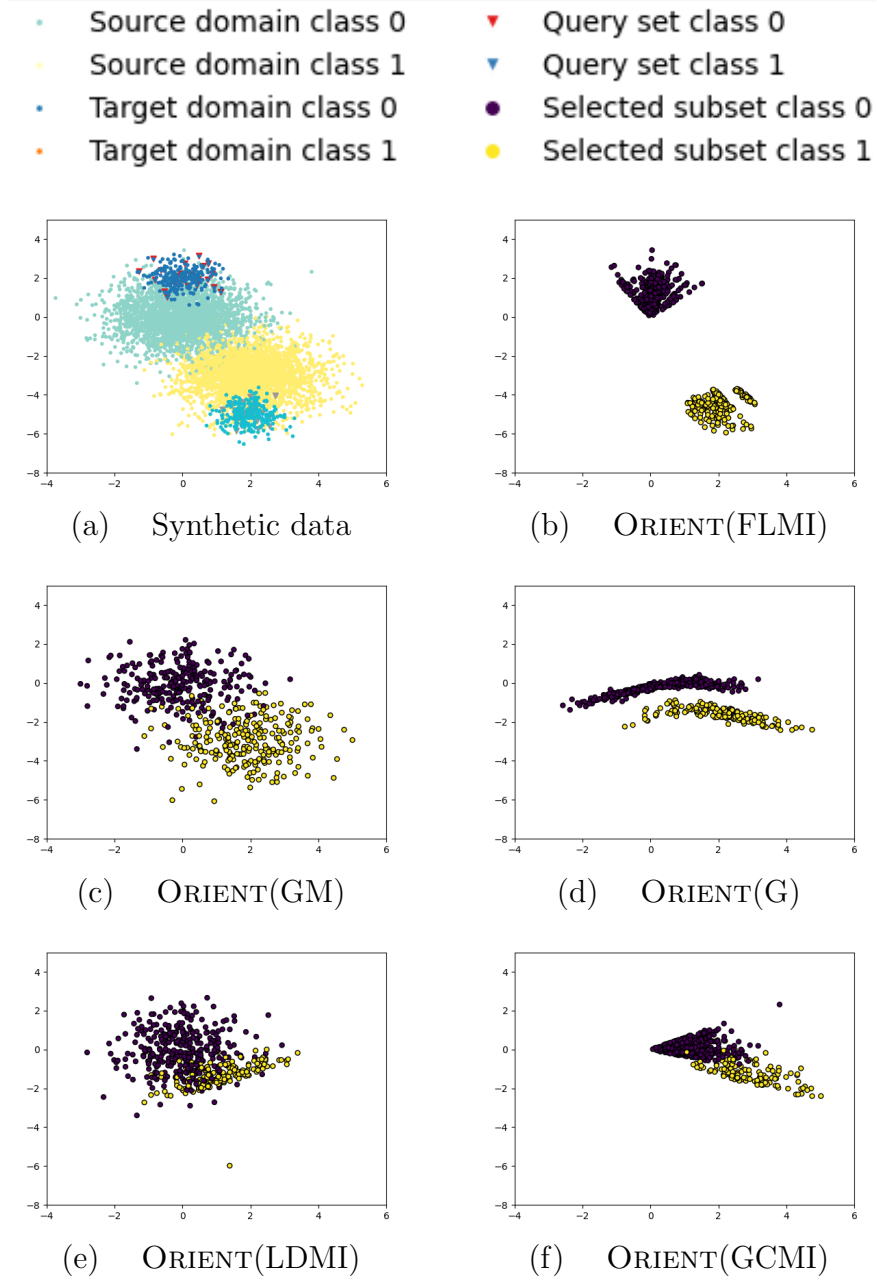


Figure D.2: Subsets selected by different instantiations of ORIENT on a synthetic dataset. (a) Synthetic data - We sample 50 examples from the target distributions for the query set. (b) ORIENT (FLMI) selects samples close to the target distribution. (c) ORIENT (GM) selects representative samples from the source domain. (d) ORIENT (G) selects samples near the decision boundaries of the source domains. (e) ORIENT(LDMI) prioritizes selection of data samples from certain classes over others (f) ORIENT(GCMI) selects samples from a source domain that are very similar and clustered together.

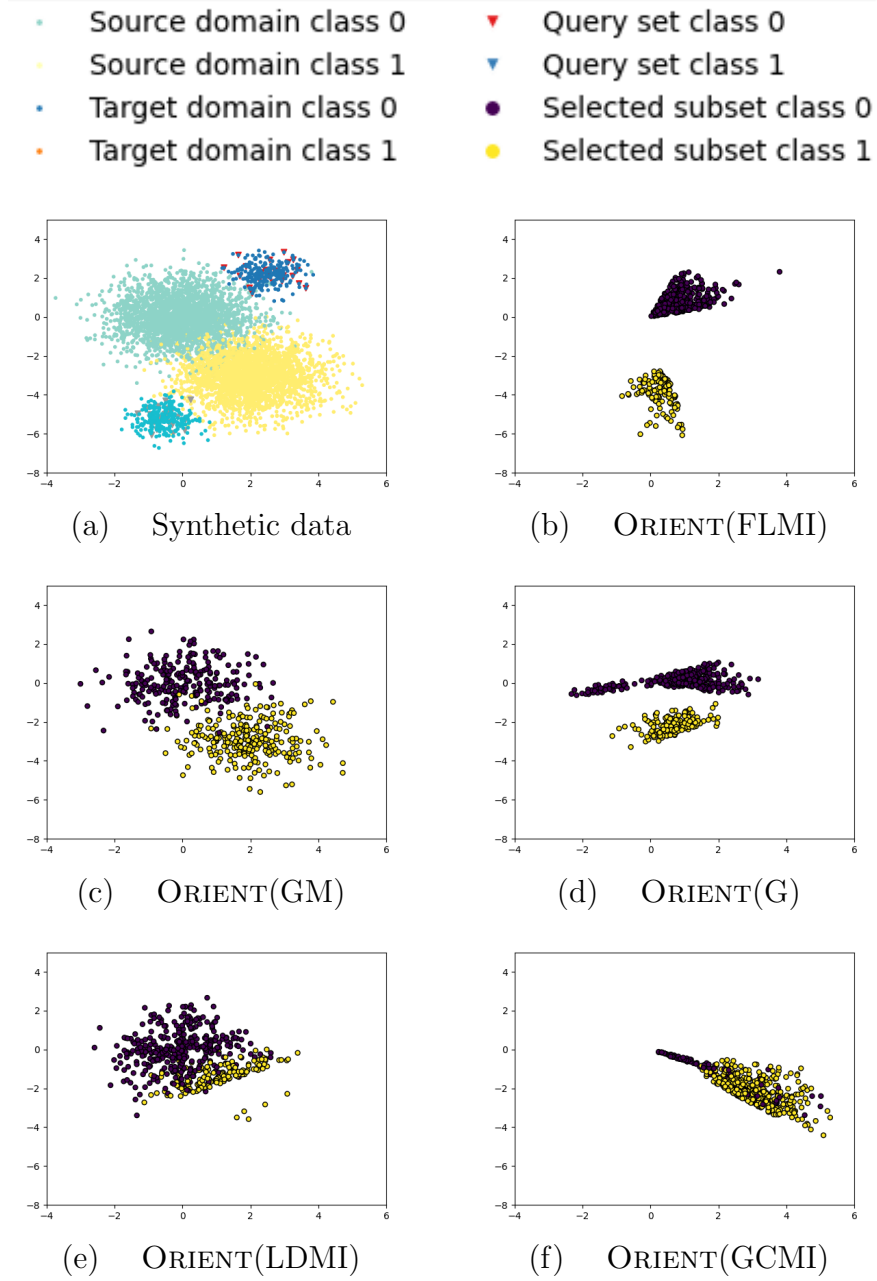


Figure D.3: Subsets selected by different instantiations of ORIENT on a synthetic dataset. (a) Synthetic data - We sample 50 examples from the target distributions for the query set. Target domain is skewed to the right for class 0 and left for class 1 as compared to the source domain. (b) ORIENT (FLMI) selects samples close to the target distribution. (c) ORIENT (GM) selects representative samples from the source domain. (d) ORIENT (G) selects samples near the decision boundaries of the source domains. (e) ORIENT(LDMI) prioritizes selection of data samples from certain classes over others (f) ORIENT(GCMI) selects samples from a source domain that are very similar and clustered together.

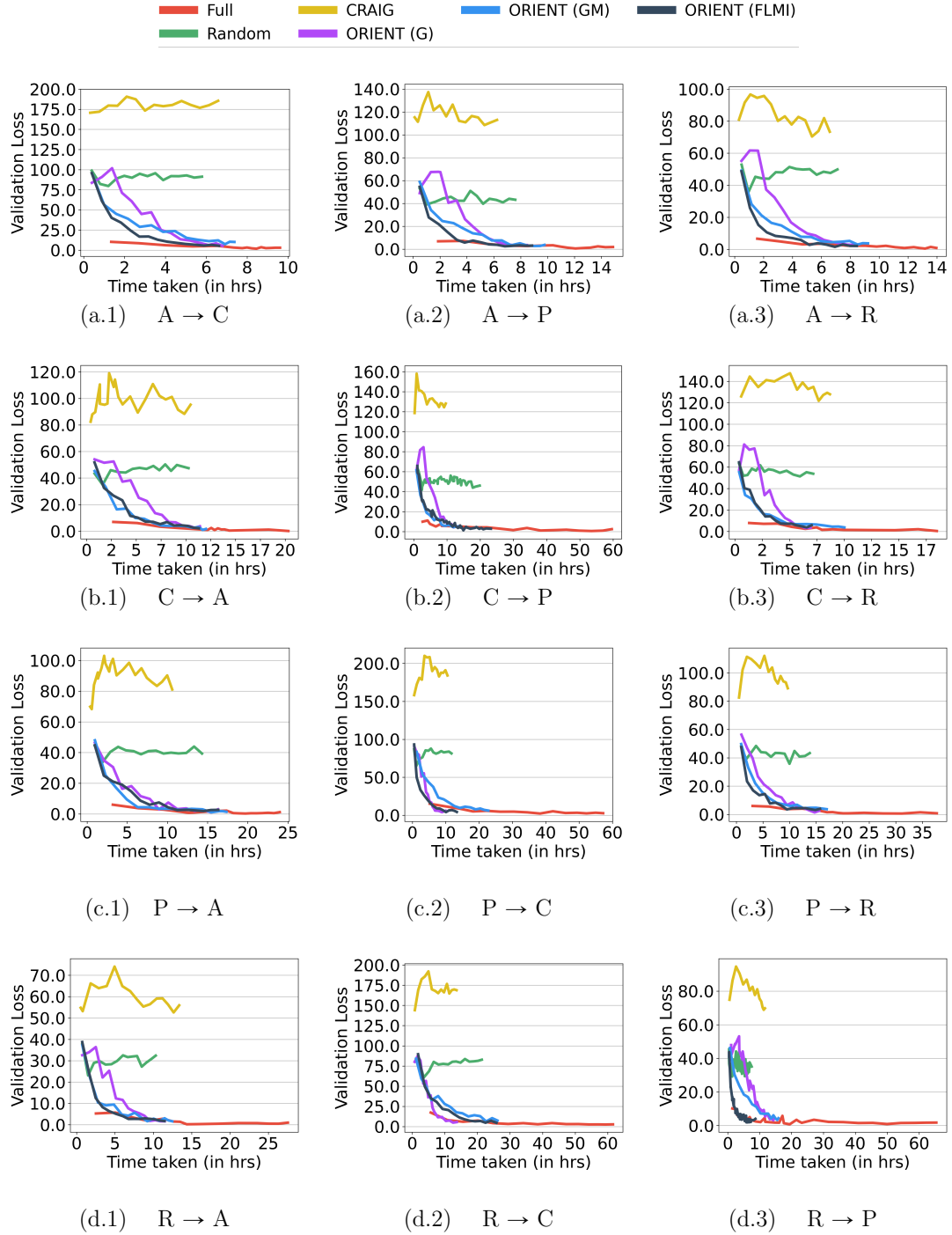


Figure D.4: Convergence curves on four domains of Office-Home dataset: Art (A), Clipart (C), Product (P), and Real World (R). X-axis presents the training time in hours and Y-axis presents the Validation loss on the target domain.

Table D.24: Speed ups achieved by different variants of ORIENT when using  $1.05\times$  the minimum validation loss as a stopping criterion for training.

	$C \rightarrow P$	$C \rightarrow R$	$R \rightarrow A$	$P \rightarrow R$	$R \rightarrow P$	$A \rightarrow C$	$C \rightarrow A$	$R \rightarrow C$	$P \rightarrow A$	$A \rightarrow P$	$A \rightarrow R$	$P \rightarrow C$	Average speedup
ORIENT (FLMI)	3.58	2.81	1.25	2.1	3.1	1.28	1.8	2.41	1.36	1.77	1.9	3.8	2.26
ORIENT (G)	4.46	2.86	1.32	2.12	1.36	1.24	1.86	4.33	1.28	1.56	1.69	4.3	2.37
ORIENT (GM)	4.49	1.84	1.56	1.83	1.25	1.22	1.74	2.39	1.27	1.34	1.55	1.68	1.85

## REFERENCES

- Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zhang (2016). Tensorflow: A system for large-scale machine learning. *CoRR abs/1605.08695*.
- Abel, D., D. Arumugam, L. Lehnert, and M. L. Littman (2018). State abstractions for lifelong reinforcement learning. In *ICML*, Volume 80 of *Proceedings of Machine Learning Research*, pp. 10–19. PMLR.
- Abiteboul, S., R. Hull, and V. Vianu (1995). *Foundations of Databases*. Addison-Wesley.
- Alkhazraji, Y., M. Frorath, M. Grützner, M. Helmert, T. Liebetraut, R. Mattmüller, M. Ortlieb, J. Seipp, T. Springenberg, P. Stahl, and J. Wülfing (2020). Pyperplan. <https://doi.org/10.5281/zenodo.3700819>.
- Altendorf, E., A. C. Restificar, and T. G. Dietterich (2005). Learning from sparse data by exploiting monotonicity constraints. In *UAI*, pp. 18–26. AUAI Press.
- Anderson, G., A. Verma, I. Dillig, and S. Chaudhuri (2020). Neurosymbolic reinforcement learning with formally verified exploration. In *NeurIPS*.
- Andre, D. and S. J. Russell (2002a). State abstraction for programmable reinforcement learning agents. In *AAAI/IAAI*, pp. 119–125. AAAI Press / The MIT Press.
- Andre, D. and S. J. Russell (2002b). State abstraction for programmable reinforcement learning agents. In *AAAI*, pp. 119–125.
- Andreas, J., D. Klein, and S. Levine (2017). Modular multitask reinforcement learning with policy sketches. In *ICML*, Volume 70 of *Proceedings of Machine Learning Research*, pp. 166–175. PMLR.
- Andrychowicz, M., D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba (2017). Hindsight experience replay. In *NeurIPS*, pp. 5048–5058.
- Anglin, P. M. and R. Gencay (1996). Semiparametric estimation of a hedonic price function. *Journal of Applied Econometrics* 11(6).
- Arpit, D., S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio, and S. Lacoste-Julien (2017). A closer look at memorization in deep networks. In *ICML*, Volume 70, pp. 233–242. PMLR.
- Ash, J. T. and R. P. Adams (2020). On warm-starting neural network training. In *NeurIPS*.

- Ash, J. T., C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal (2020). Deep batch active learning by diverse, uncertain gradient lower bounds. In *ICLR*. OpenReview.net.
- Axelrod, A., X. He, and J. Gao (2011). Domain adaptation via pseudo in-domain data selection. In *EMNLP*, pp. 355–362. ACL.
- Bacchus, F. and Q. Yang (1991). The downward refinement property. In J. Mylopoulos and R. Reiter (Eds.), *IJCAI*, pp. 286–293. Morgan Kaufmann.
- Badreddine, S., A. d’Avila Garcez, L. Serafini, and M. Spranger (2022). Logic tensor networks. *Artif. Intell.* 303, 103649.
- Bahdanau, D., K. Cho, and Y. Bengio (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.
- Banarescu, L., C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider (2013). Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL*, pp. 178–186.
- Bartley, C., W. Liu, and M. Reynolds (2016a). Effective monotone knowledge integration in kernel support vector machines. In *ADMA*, Volume 10086 of *Lecture Notes in Computer Science*, pp. 3–18.
- Bartley, C., W. Liu, and M. Reynolds (2016b). A novel technique for integrating monotone domain knowledge into the random forest classifier. In *AusDM*, Volume 170 of *CRPIT*. Australian Computer Society.
- Bartley, C., W. Liu, and M. Reynolds (2019). Enhanced random forest algorithms for partially monotone ordinal classification. In *AAAI*, pp. 3224–3231. AAAI Press.
- Battaglia, P. W., J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülgeyre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu (2018). Relational inductive biases, deep learning, and graph networks. *CoRR abs/1806.01261*.
- Ben-David, A. (1995). Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning* 19(1), 29–43.
- Ben-David, S., J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Vaughan (2010). A theory of learning from different domains. *Machine Learning* 79, 151–175.
- Ben-David, S., J. Blitzer, K. Crammer, and F. Pereira (2006). Analysis of representations for domain adaptation. In *NIPS*, pp. 137–144. MIT Press.

- Bercher, P., R. Alford, and D. Höller (2019). A survey on hierarchical planning - one abstract idea, many concrete realizations. In *IJCAI*, pp. 6267–6275. ijcai.org.
- Bilmes, J. A. (2022). Submodularity in machine learning and artificial intelligence. *CoRR abs/2202.00132*.
- Bioch, J. C. and V. Popova (2002). Monotone decision trees and noisy data. Technical report, Erasmus Research Institute of Management.
- Blaylock, N., J. Allen, and G. Ferguson (2003). *Managing Communicative Intentions with Collaborative Problem Solving*, pp. 63–84. Dordrecht: Springer Netherlands.
- Blockeel, H. and L. D. Raedt (1998). Top-down induction of first-order logical decision trees. *Artif. Intell.* 101(1-2), 285–297.
- Bonakdarpour, M., S. Chatterjee, R. F. Barber, and J. Lafferty (2018). Prediction rule reshaping. In *ICML*, Volume 80 of *Proceedings of Machine Learning Research*, pp. 629–637. PMLR.
- Bonn, J., M. Palmer, Z. Cai, and K. Wright-Bettner (2020). Spatial AMR: expanded spatial annotation in the context of a grounded minecraft corpus. In *LREC*, pp. 4883–4892. European Language Resources Association.
- Booch, G., F. Fabiano, et al. (2021). Thinking fast and slow in AI. In *AAAI*, pp. 15042–15046.
- Borsos, Z., M. Mutný, M. Tagliasacchi, and A. Krause (2021). Data summarization via bilevel optimization. *CoRR abs/2109.12534*.
- Bos, J. (2016). Expressive Power of Abstract Meaning Representations. *Computational Linguistics* 42(3), 527–535.
- Brachman, R. J. and H. J. Levesque (2004). *Knowledge Representation and Reasoning*. Elsevier.
- Brafman, R. I. and M. Tennenholtz (2002). R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.* 3, 213–231.
- Breiman, L. (1999, 10). Prediction Games and Arcing Algorithms. *Neural Computation* 11(7), 1493–1517.
- Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone (1984). *Classification and Regression Trees*. Wadsworth.

- Cabral, J., R. C. Kahlert, C. Matuszek, M. J. Witbrock, and B. Summers (2005). Converting semantic meta-knowledge into inductive bias. In *ILP*, Volume 3625 of *Lecture Notes in Computer Science*, pp. 38–50. Springer.
- Campero, A., A. Pareja, T. Klinger, J. Tenenbaum, and S. Riedel (2018). Logical rule induction and theory learning using neural theorem proving. *CoRR abs/1809.02193*.
- Cano, J. R., P. A. Gutiérrez, B. Krawczyk, M. Wozniak, and S. García (2019). Monotonic classification: An overview on algorithms, performance measures and data sets. *Neuro-computing* 341, 168–182.
- Chen, C. and S. Li (2014). Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications* 41(16), 7235–7247.
- Chen, T. and C. Guestrin (2016). Xgboost: A scalable tree boosting system. In *KDD*, pp. 785–794. ACM.
- Chu, B., V. Madhavan, O. Beijbom, J. Hoffman, and T. Darrell (2016). Best practices for fine-tuning visual classifiers to new domains. In *ECCV Workshops (3)*, Volume 9915 of *Lecture Notes in Computer Science*, pp. 435–442.
- Ciresan, D. C., U. Meier, and J. Schmidhuber (2012). Multi-column deep neural networks for image classification. In *CVPR*, pp. 3642–3649. IEEE Computer Society.
- Clodic, A., H. Cao, S. Alili, V. Montreuil, R. Alami, and R. Chatila (2008). SHARY: A supervision system adapted to human-robot interaction. In *ISER*, Volume 54 of *Springer Tracts in Advanced Robotics*, pp. 229–238. Springer.
- Cohen, N. and A. Shashua (2017). Inductive bias of deep convolutional networks through pooling geometry. In *ICLR (Poster)*. OpenReview.net.
- Cohen, W. W., F. Yang, and K. Mazaitis (2020). Tensorlog: A probabilistic database implemented using deep-learning infrastructure. *J. Artif. Intell. Res.* 67, 285–325.
- Colombo, D. and M. H. Maathuis (2014). Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research* 15(1), 3741–3782.
- Cropper, A. and S. Dumancic (2022). Inductive logic programming at 30: A new introduction. *J. Artif. Intell. Res.* 74, 765–850.
- Das, M., P. Odom, M. R. Islam, J. R. Doppa, D. Roth, and S. Natarajan (2018). Preference-guided planning: An active elicitation approach. In *AAMAS*, pp. 1921–1923.
- Das, M., N. Ramanan, J. R. Doppa, and S. Natarajan (2020). Few-shot induction of generalized logical concepts via human guidance. *Frontiers Robotics AI* 7, 122.

- Das, S., S. Natarajan, K. Roy, R. Parr, and K. Kersting (2020). Fitted q-learning for relational domains. *CoRR abs/2006.05595*.
- d’Ascoli, S., L. Sagun, G. Biroli, and J. Bruna (2019). Finding the needle in the haystack with convolutions: on the benefits of architectural bias. In *NeurIPS*, pp. 9330–9340.
- d’Avila Garcez, A. S., T. R. Besold, L. D. Raedt, P. Földiák, P. Hitzler, T. Icard, K. Kühnberger, L. C. Lamb, R. Miikkulainen, and D. L. Silver (2015). Neural-symbolic learning and reasoning: Contributions and challenges. In *AAAI Spring Symposia*. AAAI Press.
- de Campos, C. P., Y. Tong, and Q. Ji (2008). Constrained maximum likelihood learning of bayesian networks for facial action recognition. In *ECCV (3)*, Volume 5304 of *Lecture Notes in Computer Science*, pp. 168–181. Springer.
- Devin, C., D. Geng, P. Abbeel, T. Darrell, and S. Levine (2019). Compositional plan vectors. In *NeurIPS*, pp. 14963–14974.
- Devin, S. and R. Alami (2016). An implemented theory of mind to improve human-robot shared plans execution. In *HRI*, pp. 319–326. IEEE/ACM.
- Dietterich, T. G. (1998). The maxq method for hierarchical reinforcement learning. In *ICML*, pp. 118–126.
- Dietterich, T. G. (2000a). An overview of hierarchical reinforcement learning. In *International Symposium on Abstraction, Reformulation, and Approximation*. Springer.
- Dietterich, T. G. (2000b). State abstraction in maxq hierarchical reinforcement learning. In *NeurIPS*, pp. 994–1000.
- Dong, H., J. Mao, T. Lin, C. Wang, L. Li, and D. Zhou (2019). Neural logic machines. In *ICLR*.
- Dorogush, A. V., V. Ershov, and A. Gulin (2017). Catboost: gradient boosting with categorical features support. In *Workshop on ML Systems at NeurIPS*.
- Driessens, K., J. Ramon, and H. Blockeel (2001). Speeding up relational reinforcement learning through the use of an incremental first order decision tree learner. In *ECML*, Volume 2167 of *Lecture Notes in Computer Science*, pp. 97–108. Springer.
- Dua, D. and C. Graff (2017). UCI ML Repository. <http://archive.ics.uci.edu/ml>.
- Duivesteijn, W. and A. Feelders (2008). Nearest neighbour classification with monotonicity constraints. In *ECML/PKDD (1)*, Volume 5211 of *Lecture Notes in Computer Science*, pp. 301–316. Springer.

- Dzeroski, S. (2010). Relational data mining. In *Data Mining and Knowledge Discovery Handbook*, pp. 887–911. Springer.
- Dzeroski, S., L. D. Raedt, and K. Driessens (2001). Relational reinforcement learning. *Mach. Learn.* 43(1/2), 7–52.
- Edelkamp, S. and J. Hoffmann (2004). Pddl2.2: The language for the classical part of the 4th international planning competition. Technical report, Technical Report 195, University of Freiburg.
- Eppe, M., P. D. H. Nguyen, and S. Wermter (2019). From semantics to execution: Integrating action planning with reinforcement learning for robotic causal problem-solving. *Frontiers Robotics AI* 6, 123.
- Erol, K., J. A. Hendler, and D. S. Nau (1994). HTN planning: Complexity and expressivity. In *AAAI*, pp. 1123–1128. AAAI Press / The MIT Press.
- Evans, R. and E. Grefenstette (2018). Learning explanatory rules from noisy data. *J. Artif. Intell. Res.* 61, 1–64.
- Feelders, A. J. and M. Pardoel (2003). Pruning for monotone classification trees. In *IDA*, Volume 2810 of *Lecture Notes in Computer Science*, pp. 1–12. Springer.
- Feldman, D. (2020). Core-sets: Updated survey. In *Sampling Techniques for Supervised or Unsupervised Tasks*, pp. 23–44. Springer.
- Fern, A., S. W. Yoon, and R. Givan (2006). Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *J. Artif. Intell. Res.* 25, 75–118.
- FICO (2018). Explainable machine learning challenge. <https://community.fico.com/s/explainable-machine-learning-challenge>.
- Finzi, A. and T. Lukasiewicz (2006). Adaptive multi-agent programming in gtgolog. In *KI*, Volume 4314, pp. 389–403.
- Fiore, M., A. Clodic, and R. Alami (2014). On planning and task achievement modalities for human-robot collaboration. In *ISER*, Volume 109 of *Springer Tracts in Advanced Robotics*, pp. 293–306. Springer.
- Fortunato, M., M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg (2018). Noisy networks for exploration. In *ICLR (Poster)*. OpenReview.net.
- Fox, M. and D. Long (2002). Pddl+: Planning with time and metric resources. Technical report, Tech. rep. Department of Computer Science, 21/02, University of Durham, UK.

- Fox, M. and D. Long (2003). PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.* 20, 61–124.
- Freund, Y. and R. E. Schapire (1996). Experiments with a new boosting algorithm. In *ICML*, pp. 148–156. Morgan Kaufmann.
- Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Statistics* 29(5), 1189 – 1232.
- Friedman, N., L. Getoor, D. Koller, and A. Pfeffer (1999). Learning probabilistic relational models. In *IJCAI*, pp. 1300–1309. Morgan Kaufmann.
- Fu, L. and B. G. Buchanan (1985). Learning intermediate concepts in constructing a hierarchical knowledge base. In *IJCAI*, pp. 659–666. Morgan Kaufmann.
- Fujimoto, S., D. Meger, D. Precup, O. Nachum, and S. S. Gu (2022). Why should I trust you, bellman? the bellman error is a poor replacement for value error. In *ICML*, Volume 162 of *Proceedings of Machine Learning Research*, pp. 6918–6943. PMLR.
- Fujishige, S. (2005). *Submodular functions and optimization*. Elsevier.
- Fung, G., O. L. Mangasarian, and J. W. Shavlik (2002). Knowledge-based support vector machine classifiers. In *NeurIPS*, pp. 521–528. MIT Press.
- Ganin, Y. and V. S. Lempitsky (2015). Unsupervised domain adaptation by backpropagation. In *ICML*, Volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1180–1189. JMLR.org.
- Gatt, A. and E. Krahmer (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.* 61, 65–170.
- Geffner, H. and B. Bonet (2013). *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.
- Geirhos, R., C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge, and F. A. Wichmann (2018). Generalisation in humans and deep neural networks. In *NeurIPS*, pp. 7549–7561.
- Gerevini, A. and D. Long (2005). Plan constraints and preferences in pddl3. Technical report, Technical Report 2005-08-07, Department of Electronics for Automation, University of Brescia, Brescia, Italy.
- Getoor, L. and B. Tasker (Eds.) (2007). *Introduction to statistical relational learning*. MIT press.

- Ghallab, M., A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins (1998). Pddl — the planning domain definition language. *Technical Report, Tech. Rep.*.
- Ghallab, M., D. S. Nau, and P. Traverso (2004). *Automated planning - theory and practice*. Elsevier.
- Girshick, R. B., J. Donahue, T. Darrell, and J. Malik (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pp. 580–587. IEEE Computer Society.
- Givan, R., T. Dean, and M. Greig (2003). Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence* 147(1-2), 163–223.
- Glanos, C., Z. Jiang, X. Feng, P. Weng, M. Zimmer, D. Li, W. Liu, and J. Hao (2022). Neuro-symbolic hierarchical rule induction. In *ICML*, Volume 162, pp. 7583–7615.
- Goldman, R. P. and U. Kuter (2015). Measuring plan diversity: Pathologies in existing approaches and A new plan distance metric. In *AAAI*, pp. 3275–3282. AAAI Press.
- Gong, B., Y. Shi, F. Sha, and K. Grauman (2012). Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pp. 2066–2073. IEEE Computer Society.
- González, S., F. Herrera, and S. García (2015). Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Gener. Comput.* 33(4), 367–388.
- González, S., F. Herrera, and S. García (2016). Managing monotonicity in classification by a pruned adaboost. In *HAIS*, Volume 9648 of *Lecture Notes in Computer Science*, pp. 512–523. Springer.
- Goodfellow, I. J., Y. Bengio, and A. C. Courville (2016). *Deep Learning*. Adaptive computation and machine learning. MIT Press.
- Grounds, M. and D. Kudenko (2005). Combining reinforcement learning with symbolic planning. In *AAMAS III*, Volume 4865, pp. 75–86.
- Guestrin, C., D. Koller, C. Gearhart, and N. Kanodia (2003). Generalizing plans to new environments in relational mdps. In *IJCAI*, pp. 1003–1010. Morgan Kaufmann.
- Guestrin, C., R. Patrascu, and D. Schuurmans (2002). Algorithm-directed exploration for model-based reinforcement learning in factored mdps. In *ICML*, pp. 235–242. Morgan Kaufmann.
- Guo, Y. and M. Xiao (2012). Cross language text classification via subspace co-regularized multi-view learning. In *ICML*. icml.cc / Omnipress.

- Gupta, A. and R. Levin (2020). The online submodular cover problem. In *SODA*, pp. 1525–1537. SIAM.
- Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, Volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR.
- Haas, D. M., C. B. Parker, D. A. Wing, S. Parry, W. A. Grobman, B. M. Mercer, H. N. Simhan, M. K. Hoffman, R. M. Silver, P. Wadhwa, J. D. Iams, M. A. Koch, S. N. Caritis, R. J. Wapner, M. S. Esplin, M. A. Elovitz, T. Foroud, A. M. Peaceman, G. R. Saade, M. Willinger, and U. M. Reddy (2015). A description of the methods of the nulliparous pregnancy outcomes study: monitoring mothers-to-be (nuMoM2b). *American journal of obstetrics and gynecology* 212(4), 539.e1–539.e24.
- Hayes, A. L., M. Das, P. Odom, and S. Natarajan (2017). User friendly automatic construction of background knowledge: Mode construction from ER diagrams. In *K-CAP*, pp. 30:1–30:8. ACM.
- He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *CVPR*, pp. 770–778. IEEE Computer Society.
- Hedderson, M. M., J. A. Darbinian, and A. Ferrara (2010). Disparities in the risk of gestational diabetes by race-ethnicity and country of birth. *Paediatric and Perinatal Epidemiology* 24(5), 441–448.
- Hedegaard, L., O. A. Sheikh-Omar, and A. Iosifidis (2021). Supervised domain adaptation: A graph embedding perspective and a rectified experimental protocol. *IEEE Trans. Image Process.* 30, 8619–8631.
- Helsper, E. M., L. C. van der Gaag, and F. Groenendaal (2004). Designing a procedure for the acquisition of probability constraints for bayesian networks. In *EKAU*, Volume 3257 of *Lecture Notes in Computer Science*, pp. 280–292. Springer.
- Hershey, J. R., S. J. Rennie, P. A. Olsen, and T. T. Kristjansson (2010). Super-human multi-talker speech recognition: A graphical modeling approach. *Comput. Speech Lang.* 24(1), 45–66.
- Hessel, M., J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver (2018). Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, pp. 3215–3222. AAAI Press.
- Höller, D., G. Behnke, P. Bercher, S. Biundo, H. Fiorino, D. Pellier, and R. Alford (2020). HDDL: an extension to PDDL for expressing hierarchical planning problems. In *AAAI*, pp. 9883–9891.

- Hu, Z., X. Ma, Z. Liu, E. H. Hovy, and E. P. Xing (2016). Harnessing deep neural networks with logic rules. In *ACL (1)*. The Association for Computer Linguistics.
- Huang, D., S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles (2019). Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *CVPR*, pp. 8565–8574. Computer Vision Foundation / IEEE.
- Huang, S. and S. Ontañón (2022). A closer look at invalid action masking in policy gradient algorithms. In *FLAIRS*.
- Igl, M., G. Farquhar, J. Luketina, W. Boehmer, and S. Whiteson (2021). Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations*.
- Ilghami, O. (2006). Documentation for jshop2. *Tech Report*.
- Illanes, L., X. Yan, R. T. Icarte, and S. A. McIlraith (2020). Symbolic plans as high-level instructions for reinforcement learning. *ICAPS*, 540–550.
- Iyer, R. K., N. Khargonkar, J. A. Bilmes, and H. Asnani (2022). Generalized submodular information measures: Theoretical properties, examples, optimization algorithms, and applications. *IEEE Trans. Inf. Theory* 68(2), 752–781.
- Jackson, E., D. E. Appelt, J. Bear, R. C. Moore, and A. Podlozny (1991). A template matcher for robust NL interpretation. In *HLT*. Morgan Kaufmann.
- Janisch, J., T. Pevný, and V. Lisý (2021). Symbolic relational deep reinforcement learning based on graph neural networks. *RL4RealLife @ ICML2021*.
- Jayannavar, P., A. Narayan-Chen, and J. Hockenmaier (2020). Learning to execute instructions in a minecraft dialogue. In *ACL*, pp. 2589–2602.
- Jiang, Y., F. Yang, S. Zhang, and P. Stone (2019). Task-motion planning with reinforcement learning for adaptable mobile service robots. In *IROS*, pp. 7529–7534.
- Jiang, Z. and S. Luo (2019, 09–15 Jun). Neural logic reinforcement learning. In *ICML*, Volume 97, pp. 3110–3119. PMLR.
- Jiang, Z., P. Minervini, M. Jiang, and T. Rocktäschel (2021). Grid-to-graph: Flexible spatial relational inductive biases for reinforcement learning. In *AAMAS*, pp. 674–682. ACM.
- Johnson, M., K. Hofmann, T. Hutton, and D. Bignell (2016). The malmo platform for ai experimentation. In *IJCAI*.
- Kaggle (2020). State of data science and machine learning survey. <https://www.kaggle.com/kaggle-survey-2020>.

- Kaggle (2021). State of data science and machine learning survey. <https://www.kaggle.com/kaggle-survey-2021>.
- Kaggle (2022). State of data science and machine learning survey. <https://www.kaggle.com/kaggle-survey-2022>.
- Kanervisto, A., S. Milani, K. Ramanauskas, N. Topin, Z. Lin, J. Li, J. Shi, D. Ye, Q. Fu, W. Yang, W. Hong, Z. Huang, H. Chen, G. Zeng, Y. Lin, V. Micheli, E. Alonso, F. Fleuret, A. Nikulin, Y. Belousov, O. Svidchenko, and A. Shpilman (2021). Miner1 diamond 2021 competition: Overview, results, and lessons learned. In *NeurIPS (Competition and Demos)*, Volume 176 of *Proceedings of Machine Learning Research*, pp. 13–28. PMLR.
- Karanam, A., A. L. Hayes, H. Kokel, D. M. Haas, P. Radivojac, and S. Natarajan (2021). A probabilistic approach to extract qualitative knowledge for early prediction of gestational diabetes. In *AIME*, Volume 12721 of *Lecture Notes in Computer Science*, pp. 497–502. Springer.
- Kaur, N., G. Kunapuli, S. Joshi, K. Kersting, and S. Natarajan (2019). Neural networks for relational data. In *ILP*, Volume 11770 of *Lecture Notes in Computer Science*, pp. 62–71. Springer.
- Kaur, N., G. Kunapuli, and S. Natarajan (2020). Non-parametric learning of lifted restricted boltzmann machines. *Int. J. Approx. Reason.* 120, 33–47.
- Kaushal, V., R. K. Iyer, S. Kothawade, R. Mahadev, K. Doctor, and G. Ramakrishnan (2019). Learning from less data: A unified data subset selection and active learning framework for computer vision. In *WACV*, pp. 1289–1299. IEEE.
- Kautz, H. A. (2022). The third AI summer: AAAI robert s. engelmore memorial lecture. *AI Mag.* 43(1), 93–104.
- Kazemi, S. M. and D. Poole (2018). Relnn: A deep neural model for relational learning. In *AAAI*, pp. 6367–6375. AAAI Press.
- Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *NeurIPS*, pp. 3146–3154.
- Kersting, K. and L. D. Raedt (2001). Towards combining inductive logic programming with bayesian networks. In *ILP*, Volume 2157 of *Lecture Notes in Computer Science*, pp. 118–131. Springer.
- Killamsetty, K., G. S. Abhishek, Aakriti, A. V. Evfimievski, L. Popa, G. Ramakrishnan, and R. Iyer (2022). AUTOMATA: Gradient based data subset selection for compute-efficient hyper-parameter tuning. In *NeurIPS*.

- Killamsetty, K., D. Sivasubramanian, G. Ramakrishnan, A. De, and R. K. Iyer (2021). GRAD-MATCH: gradient matching based data subset selection for efficient deep model training. In *ICML*, Volume 139, pp. 5464–5474. PMLR.
- Killamsetty, K., D. Sivasubramanian, G. Ramakrishnan, and R. K. Iyer (2021). GLISTER: generalization based data subset selection for efficient and robust learning. In *AAAI*, pp. 8110–8118. AAAI Press.
- Killamsetty, K., X. Zhao, F. Chen, and R. K. Iyer (2021). RETRIEVE: coreset selection for efficient and robust semi-supervised learning. In *NeurIPS*, pp. 14488–14501.
- Kim, M. and I. Han (2003). The discovery of experts’ decision rules from qualitative bankruptcy data using genetic algorithms. *Expert Systems with Applications* 25(4), 637–646.
- Kimura, D., M. Ono, S. Chaudhury, R. Kohita, A. Wachi, D. J. Agravante, M. Tatsubori, A. Munawar, and A. Gray (2021). Neuro-symbolic reinforcement learning with first-order logic. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 3505–3511. Association for Computational Linguistics.
- Kirchhoff, K. and J. A. Bilmes (2014). Submodularity for data selection in machine translation. In *EMNLP*, pp. 131–141. ACL.
- Köhn, A., J. Wichlacz, C. Schäfer, Á. Torralba, J. Hoffmann, and A. Koller (2020). MC-saar-instruct: a platform for Minecraft instruction giving agents. In *SIGDIAL*, pp. 53–56.
- Kokel, H., M. Das, R. Islam, J. Bonn, J. Cai, S. Dan, A. Narayan-Chen, P. Jayannavar, J. R. Doppa, J. Hockenmaier, S. Natarajan, M. Palmer, and D. Roth (2021). Human-guided collaborative problem solving: A natural language based framework. *ICAPS (Demo Track)*.
- Kokel, H., M. Das, R. Islam, J. Bonn, J. Cai, S. Dan, A. Narayan-Chen, P. Jayannavar, J. R. Doppa, J. Hockenmaier, S. Natarajan, M. Palmer, and D. Roth (2022). Lara – human-guided collaborative problem solver: Effective integration of learning, reasoning and communication. *The Tenth Annual Conference on Advances in Cognitive Systems (ACS)*.
- Kokel, H., A. Manoharan, S. Natarajan, B. Ravindran, and P. Tadepalli (2021a). Dynamic probabilistic logic models for effective abstractions in RL. *StarAI Workshop @ AAAI*.
- Kokel, H., A. Manoharan, S. Natarajan, B. Ravindran, and P. Tadepalli (2021b, May). RePReL: Integrating relational planning and reinforcement learning for effective abstraction. *ICAPS* 31(1), 533–541.

- Kokel, H., S. Natarajan, B. Ravindran, and P. Tadepalli (2022a). Dynamic probabilistic logic models for effective abstractions in RL (Abstract). *5th Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*.
- Kokel, H., S. Natarajan, B. Ravindran, and P. Tadepalli (2022b). RePreL: a unified framework for integrating relational planning and reinforcement learning for effective abstraction in discrete and continuous domains. *Neural Computing and Applications*.
- Kokel, H., P. Odom, S. Yang, and S. Natarajan (2020). A unified framework for knowledge intensive gradient boosting: Leveraging human experts for noisy sparse domains. In *AAAI*, Volume 34, pp. 4460–4468. AAAI Press.
- Kokel, H., N. Prabhakar, B. Ravindran, E. Blasch, P. Tadepalli, and S. Natarajan (2022). Hybrid deep repel: Integrating relational planning and reinforcement learning for information fusion. *IEEE 25th International Conference on Information Fusion (FUSION)*.
- Koller, D. and N. Friedman (2009). *Probabilistic Graphical Models - Principles and Techniques*. MIT Press.
- Konidaris, G. (2019). On the necessity of abstraction. *Current Opinion in Behavioral Sciences* 29, 1–7.
- Koniusz, P., Y. Tas, and F. Porikli (2017). Domain adaptation by mixture of alignments of second-or higher-order scatter tensors. In *CVPR*, pp. 7139–7148. IEEE Computer Society.
- Konstas, I., S. Iyer, M. Yatskar, Y. Choi, and L. Zettlemoyer (2017). Neural AMR: sequence-to-sequence models for parsing and generation. In *ACL (1)*, pp. 146–157.
- Kothawade, S., N. Beck, K. Killamsetty, and R. K. Iyer (2021). SIMILAR: submodular information measures based active learning in realistic scenarios. In *NeurIPS*, pp. 18685–18697.
- Kothawade, S., V. Kaushal, G. Ramakrishnan, J. A. Bilmes, and R. K. Iyer (2022). PRISM: a rich class of parameterized submodular information measures for guided subset selection. *AAAI* 36.
- Krishnaswamy, N., P. Narayana, R. Bangar, K. Rim, D. Patil, D. G. McNeely-White, J. Ruiz, B. A. Draper, J. R. Beveridge, and J. Pustejovsky (2020). Diana’s world: A situated multimodal interactive agent. In *AAAI*, pp. 13618–13619. AAAI Press.
- Kulkarni, T. D., K. Narasimhan, A. Saeedi, and J. Tenenbaum (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NeurIPS*, pp. 3675–3683.
- Lake, B. M., R. Salakhutdinov, and J. B. Tenenbaum (2015). Human-level concept learning through probabilistic program induction. *Science* 350(6266), 1332–1338.

- Lamb, L. C., A. S. d’Avila Garcez, M. Gori, M. O. R. Prates, P. H. C. Avelar, and M. Y. Vardi (2020). Graph neural networks meet neural-symbolic computing: A survey and perspective. In *IJCAI*, pp. 4877–4884.
- Lemaignan, S., M. Warnier, E. A. Sisbot, A. Clodic, and R. Alami (2017). Artificial cognition for social human-robot interaction: An implementation. *Artif. Intell.* *247*, 45–69.
- Li, A., T. Luo, Z. Lu, T. Xiang, and L. Wang (2019). Large-scale few-shot learning: Knowledge transfer with class hierarchy. In *CVPR*, pp. 7212–7220. Computer Vision Foundation / IEEE.
- Li, L., T. J. Walsh, and M. L. Littman (2006). Towards a unified theory of state abstraction for mdps. In *AI&M*.
- Li, R., A. Jabri, T. Darrell, and P. Agrawal (2020). Towards practical multi-object manipulation using relational reinforcement learning. In *ICRA*, pp. 4051–4058. IEEE.
- Liu, M. and O. Tuzel (2016). Coupled generative adversarial networks. In *NeurIPS*, pp. 469–477.
- Lloyd, J. W. (1987). *Foundations of Logic Programming, 2nd Edition*. Springer.
- Lovász, L. (1982). Submodular functions and convexity. In *ISMP*, pp. 235–257. Springer.
- Lyle, C., M. Rowland, and W. Dabney (2022). Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations*.
- Lyu, D., F. Yang, B. Liu, and S. Gustafson (2019). SDRL: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *AAAI*, pp. 2970–2977. AAAI Press.
- Makino, K., T. Suda, K. Yano, and T. Ibaraki (1996). Data analysis by positive decision trees. In *CODAS*, pp. 257–264. World Scientific.
- Manhaeve, R., S. Dumancic, A. Kimmig, T. Demeester, and L. D. Raedt (2018). Deep-problog: Neural probabilistic logic programming. In *NeurIPS*, pp. 3753–3763.
- Mason, L., J. Baxter, P. L. Bartlett, and M. R. Frean (1999). Boosting algorithms as gradient descent. In *NIPS*, pp. 512–518. The MIT Press.
- McDermott, D. V. (2000). The 1998 AI planning systems competition. *AI Mag.* *21*(2), 35–55.
- Minoux, M. (1978). Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pp. 234–243. Springer.

- Mirzasoleiman, B., J. A. Bilmes, and J. Leskovec (2020). Coresets for data-efficient training of machine learning models. In *ICML*, Volume 119, pp. 6950–6960. PMLR.
- Mirzasoleiman, B., K. Cao, and J. Leskovec (2020). Coresets for robust training of deep neural networks against noisy labels.
- Mitchener, L., D. Tuckey, M. Crosby, and A. Russo (2022). Detect, understand, act: A neuro-symbolic hierarchical reinforcement learning framework. Volume 111, pp. 1523–1549.
- Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller (2013). Playing atari with deep reinforcement learning. *CoRR abs/1312.5602*.
- Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis (2015). Human-level control through deep reinforcement learning. *Nature* 518(7540), 529–533.
- Morsing, L. H., O. A. Sheikh-Omar, and A. Iosifidis (2020). Supervised domain adaptation using graph embedding. In *ICPR*, pp. 7841–7847. IEEE.
- Motiiian, S., Q. Jones, S. M. Iranmanesh, and G. Doretto (2017). Few-shot adversarial domain adaptation. In *NIPS*, pp. 6670–6680.
- Motiiian, S., M. Piccirilli, D. A. Adjeroh, and G. Doretto (2017). Unified deep supervised domain adaptation and generalization. In *ICCV*, pp. 5716–5726. IEEE Computer Society.
- Muandet, K., D. Balduzzi, and B. Schölkopf (2013). Domain generalization via invariant feature representation. In *ICML (1)*, Volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 10–18. JMLR.org.
- Muggleton, S. and C. Feng (1990). Efficient induction of logic programs. In *New Generation Computing*. Academic Press.
- Muggleton, S. H. (1992). Inductive logic programming. *Morgan Kaufmann* (38).
- Muggleton, S. H. (1995). Inverse entailment and prolog. *New Gener. Comput.* 13(3&4), 245–286.
- Muggleton, S. H. and L. D. Raedt (1994). Inductive logic programming: Theory and methods. *J. Log. Program.* 19/20, 629–679.
- Murphy, K. P. (2002). *Dynamic Bayesian networks: Representation, Inference and Learning*. Ph. D. thesis.

- Narayan-Chen, A., P. Jayannavar, and J. Hockenmaier (2019). Collaborative dialogue in minecraft. In *ACL*.
- Narayan-Chen, A. Y. (2020). *Towards collaborative dialogue in Minecraft*. Ph. D. thesis, University of Illinois at Urbana-Champaign.
- Natarajan, S. and E. E. Altendorf (2005). First order conditional influence language. Technical report, Technical Report CS05-30-01, September 23, School of EECS, Oregon State University, USA.
- Natarajan, S., P. Tadepalli, E. Altendorf, T. G. Dietterich, A. Fern, and A. C. Restificar (2005). Learning first-order probabilistic models with combining rules. In *ICML*, Volume 119 of *ACM International Conference Proceeding Series*, pp. 609–616. ACM.
- Natarajan, S., P. Tadepalli, T. G. Dietterich, and A. Fern (2008). Learning first-order probabilistic models with combining rules. *Ann. Math. Artif. Intell.* 54(1-3), 223–256.
- Nau, D. S., T. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman (2003). SHOP2: an HTN planning system. *J. Artif. Intell. Res.* 20, 379–404.
- Nau, D. S., Y. Cao, A. Lotem, and H. Muñoz-Avila (1999). SHOP: simple hierarchical ordered planner. In *IJCAI*, pp. 968–975. Morgan Kaufmann.
- Neville, J. and D. D. Jensen (2004). Dependency networks for relational data. In *ICDM*, pp. 170–177. IEEE Computer Society.
- Nilsson, M. and T. Ziemke (2007). Information fusion: a decision support perspective. In *FUSION*, pp. 1–8. IEEE.
- Nitti, D., V. Belle, T. D. Laet, and L. D. Raedt (2015). Sample-based abstraction for hybrid relational mdps. In *European Workshop on Reinforcement Learning*.
- Odom, P. and S. Natarajan (2018). Human-guided learning for probabilistic logic models. *Frontiers Robotics AI* 5, 56.
- Olson, R. S., W. L. Cava, Z. Mustahsan, A. Varik, and J. H. Moore (2018). Data-driven advice for applying machine learning to bioinformatics problems. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2018: Proceedings of the Pacific Symposium*, pp. 192–203. World Scientific.
- Parr, R. and S. J. Russell (1998). Reinforcement learning with hierarchies of machines. In *NeurIPS*.
- Patel, V. M., R. Gopalan, R. Li, and R. Chellappa (2015). Visual domain adaptation: A survey of recent advances. *IEEE Signal Process. Mag.* 32(3), 53–69.

- Pazzani, M. J., C. Brunk, and G. Silverstein (1991). A knowledge-intensive approach to learning relational concepts. In *ML*, pp. 432–436. Morgan Kaufmann.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.* 12, 2825–2830.
- Peng, X., C. Wang, D. Gildea, and N. Xue (2017). Addressing the data sparsity issue in neural AMR parsing. In *EACL (1)*, pp. 366–375.
- Picado, J., A. Termehchy, A. Fern, S. Pathak, P. Ilango, and J. Davis (2021). Scalable and usable relational learning with automatic language bias. In *SIGMOD Conference*, pp. 1440–1451. ACM.
- Plappert, M., M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR abs/1802.09464*.
- Potharst, R. and J. C. Bioch (1999). A decision tree algorithm for ordinal classification. In *IDA*, Volume 1642 of *Lecture Notes in Computer Science*, pp. 187–198. Springer.
- Potharst, R. and A. J. Feelders (2002). Classification trees for problems with monotonicity constraints. *SIGKDD Explor.* 4(1), 1–10.
- Quinlan, J. R. (1990). Learning logical definitions from relations. *Mach. Learn.* 5, 239–266.
- Raedt, L. D. (1997). Logical settings for concept-learning. *Artif. Intell.* 95(1), 187–201.
- Raedt, L. D. (2008). Logical and relational learning. In *SBIA*, Volume 5249 of *Lecture Notes in Computer Science*, pp. 1. Springer.
- Raedt, L. D. and K. Kersting (2008). Probabilistic inductive logic programming. In *Probabilistic Inductive Logic Programming*, Volume 4911 of *Lecture Notes in Computer Science*, pp. 1–27.
- Raedt, L. D., K. Kersting, S. Natarajan, and D. Poole (2016). Statistical relational artificial intelligence: Logic, probability, and computation.
- Ravindran, B. and A. G. Barto (2003). Smdp homomorphisms: An algebraic approach to abstraction in semi markov decision processes. In *IJCAI*, pp. 1011–1018.

- Richardson, M. and P. M. Domingos (2006). Markov logic networks. *Mach. Learn.* 62(1-2), 107–136.
- Riegel, R., A. G. Gray, F. P. S. Luus, N. Khan, N. Makondo, I. Y. Akhalwaya, H. Qian, R. Fagin, F. Barahona, U. Sharma, S. Ikbali, H. Karanam, S. Neelam, A. Likhyan, and S. K. Srivastava (2020). Logical neural networks. *CoRR abs/2006.13155*.
- Robertson, T., F. T. Wright, and R. Dykstra (1988). *Order restricted statistical inference*. New York: Wiley.
- Rocktäschel, T. and S. Riedel (2017). End-to-end differentiable proving. In *NIPS*, pp. 3788–3800.
- Saenko, K., B. Kulis, M. Fritz, and T. Darrell (2010). Adapting visual category models to new domains. In *ECCV (4)*, Volume 6314 of *Lecture Notes in Computer Science*, pp. 213–226. Springer.
- Saito, K., D. Kim, S. Sclaroff, T. Darrell, and K. Saenko (2019). Semi-supervised domain adaptation via minimax entropy. pp. 8049–8057.
- Salge, C., M. C. Green, R. Canaan, F. Skwarski, R. Fritsch, A. Brightmoore, S. Ye, C. Cao, and J. Togelius (2020). The AI settlement generation challenge in minecraft. *Künstliche Intell.* 34(1).
- Sammut, C. and R. B. Banerji (1986). Learning concepts by asking questions (pp. 167-191). *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.
- Schwartz, R., J. Dodge, N. Smith, and O. Etzioni (2020). Green ai. *Communications of the ACM* 63, 54 – 63.
- Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pp. 618–626. IEEE Computer Society.
- Sen, P., B. W. S. R. de Carvalho, R. Riegel, and A. G. Gray (2022). Neuro-symbolic inductive logic programming with logical neural networks. In *AAAI*, pp. 8212–8219.
- Sener, O. and S. Savarese (2018). Active learning for convolutional neural networks: A core-set approach. In *ICLR*. OpenReview.net.
- Serafini, L., I. Donadello, and A. S. d’Avila Garcez (2017). Learning and reasoning in logic tensor networks: theory and application to semantic image interpretation. In *SAC*, pp. 125–130. ACM.

- Shah, R., S. H. Wang, C. Wild, S. Milani, A. Kanervisto, V. G. Goecks, N. R. Waytowich, D. Watkins-Valls, B. Prakash, E. Mills, D. Garg, A. Fries, A. Souly, J. S. Chan, D. del Castillo, and T. Lieberum (2021). Retrospective on the 2021 miner1 BASALT competition on learning from human feedback. In *NeurIPS (Competition and Demos)*, Volume 176 of *Proceedings of Machine Learning Research*, pp. 259–272. PMLR.
- Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference* 90(2), 227–244.
- Shindo, H., D. S. Dhimi, and K. Kersting (2021). Neuro-symbolic forward reasoning. *CoRR abs/2110.09383*.
- Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis (2016). Mastering the game of go with deep neural networks and tree search. *Nat.* 529(7587), 484–489.
- Silver, D., T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* 362(6419), 1140–1144.
- Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. P. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis (2017). Mastering the game of go without human knowledge. *Nat.* 550(7676), 354–359.
- Singh, A. (2021). CLDA: contrastive learning for semi-supervised domain adaptation. In *NeurIPS*, pp. 5089–5101.
- Sorg, J. and S. Singh (2009). Transfer via soft homomorphisms. In *AAMAS (2)*, pp. 741–748. IFAAMAS.
- Sourek, G., V. Aschenbrenner, F. Zelezný, S. Schockaert, and O. Kuzelka (2018). Lifted relational neural networks: Efficient learning of latent relational structures. *J. Artif. Intell. Res.* 62, 69–100.
- Sourek, G., S. Manandhar, F. Zelezný, S. Schockaert, and O. Kuzelka (2016). Learning predictive categories using lifted relational neural networks. In *ILP*, Volume 10326 of *Lecture Notes in Computer Science*, pp. 108–119. Springer.
- Spirtes, P. and C. Glymour (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review* 9(1), 62–72.

- Srinivasan, A. (1999). The aleph manual. <https://www.cs.ox.ac.uk/activities/programinduction/Aleph/aleph>.
- Star-AI (2010). Statistical relational AI (Star-AI) workshop at AAAI. <https://www.biostat.wisc.edu/~natarasr/starAI/starai.html>.
- Sutskever, I., O. Vinyals, and Q. V. Le (2014). Sequence to sequence learning with neural networks. In *NeurIPS*, pp. 3104–3112.
- Sutton, R. S. and A. G. Barto (1998). *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press.
- Sutton, R. S., D. Precup, and S. Singh (1998). Intra-option learning about temporally abstract actions. In *ICML*, pp. 556–564. Morgan Kaufmann.
- Tadepalli, P., R. Givan, and K. Driessens (2004). Relational reinforcement learning: An overview. In *ICML workshop on relational reinforcement learning*, pp. 1–9.
- Taskar, B., P. Abbeel, and D. Koller (2002). Discriminative probabilistic models for relational data. In *UAI*, pp. 485–492. Morgan Kaufmann.
- Tausend, B. (1994). Biases and their effects in inductive logic programming. In *ECML*, Volume 784 of *Lecture Notes in Computer Science*, pp. 431–434. Springer.
- Tiwari, R., K. Killamsetty, R. K. Iyer, and P. Shenoy (2022). GCR: gradient coreset based replay buffer selection for continual learning.
- Tong, Y. and Q. Ji (2008). Learning bayesian networks with qualitative constraints. In *CVPR*. IEEE Computer Society.
- Torralba, A. and A. A. Efros (2011). Unbiased look at dataset bias. In *CVPR*, pp. 1521–1528. IEEE Computer Society.
- Towell, G. G. and J. W. Shavlik (1994). Knowledge-based artificial neural networks. *Artificial Intelligence* 70(1-2), 119–165.
- Tzeng, E., J. Hoffman, T. Darrell, and K. Saenko (2015). Simultaneous deep transfer across domains and tasks. In *ICCV*, pp. 4068–4076. IEEE Computer Society.
- Tzeng, E., J. Hoffman, K. Saenko, and T. Darrell (2017). Adversarial discriminative domain adaptation. In *CVPR*, pp. 2962–2971. IEEE Computer Society.
- Ullman, J. D. (1988). *Principles of Database and Knowledge-Base Systems, Volume I*, Volume 14 of *Principles of computer science series*. Computer Science Press.

- van de Kamp, R., A. Feelders, and N. Barile (2009). Isotonic classification trees. In *IDA*, Volume 5772 of *Lecture Notes in Computer Science*, pp. 405–416. Springer.
- van Hasselt, H., A. Guez, and D. Silver (2016). Deep reinforcement learning with double q-learning. In *AAAI*, pp. 2094–2100.
- Venkateswara, H., J. Eusebio, S. Chakraborty, and S. Panchanathan (2017). Deep hashing network for unsupervised domain adaptation. In *CVPR*, pp. 5385–5394. IEEE Computer Society.
- Vlachos, P. and M. Meyer (2005). Statlib datasets archive. <http://lib.stat.cmu.edu/datasets>.
- Walsh, T. J., L. Li, and M. L. Littman (2006). Transferring state abstractions between mdps. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*.
- Wang, M. and W. Deng (2018). Deep visual domain adaptation: A survey. *Neurocomputing* 312, 135–153.
- Wang, Z., T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas (2016). Dueling network architectures for deep reinforcement learning. In *ICML*, Volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1995–2003. JMLR.org.
- Watkins, C. J. C. H. (1989). Learning from delayed rewards. *PhD thesis, Cambridge University, Cambridge, England*.
- Wei, K., R. K. Iyer, and J. A. Bilmes (2015). Submodularity in data subset selection and active learning. In *ICML*, Volume 37, pp. 1954–1963. JMLR.org.
- Wei, K., Y. Liu, K. Kirchhoff, C. D. Bartels, and J. A. Bilmes (2014). Submodular subset selection for large-scale speech training data. In *ICASSP*, pp. 3311–3315. IEEE.
- Wei, K., Y. Liu, K. Kirchhoff, and J. A. Bilmes (2014). Unsupervised submodular subset selection for speech data. In *ICASSP*, pp. 4107–4111. IEEE.
- Wellman, M. P. (1990). Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence* 44(3), 257–303.
- Wichlacz, J., A. Torralba, and J. Hoffmann (2019). Construction-planning models in minecraft. *ICAPS workshop on HPlan*.
- Winograd, T. (1972). Understanding natural language. *Cognitive Psychology* 3(1), 1–191.
- Winston, P. H. (1970). Learning structural descriptions from examples.

- Xu, D., J. Li, M. Zhu, M. Zhang, and G. Zhou (2020). Improving AMR parsing with sequence-to-sequence pre-training. In *EMNLP (1)*, pp. 2501–2511. Association for Computational Linguistics.
- Xu, X., X. Zhou, R. Venkatesan, G. Swaminathan, and O. Majumder (2019). d-sne: Domain adaptation using stochastic neighborhood embedding. In *CVPR*, pp. 2497–2506. Computer Vision Foundation / IEEE.
- Yang, F., D. Lyu, B. Liu, and S. Gustafson (2018). PEORL: integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. pp. 4860–4866.
- Yang, F., Z. Yang, and W. W. Cohen (2017). Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, pp. 2319–2328.
- Yang, S. and S. Natarajan (2013). Knowledge intensive learning: Combining qualitative constraints with causal independence for parameter learning in probabilistic models. In *ECML/PKDD (2)*, Volume 8189 of *Lecture Notes in Computer Science*, pp. 580–595. Springer.
- Yang, Z., A. Ishay, and J. Lee (2020). Neurasp: Embracing neural networks into answer set programming. In *IJCAI*, pp. 1755–1762. ijcai.org.
- Yao, T., Y. Pan, C. Ngo, H. Li, and T. Mei (2015). Semi-supervised domain adaptation with subspace learning for visual recognition. In *CVPR*, pp. 2142–2150. IEEE Computer Society.
- Yet, B., Z. B. Perkins, T. E. Rasmussen, N. R. M. Tai, and D. W. R. Marsh (2014). Combining data and meta-analysis to build bayesian networks for clinical decision support. *J. Biomed. Informatics* 52, 373–385.
- Yosinski, J., J. Clune, Y. Bengio, and H. Lipson (2014). How transferable are features in deep neural networks? In *NeurIPS*, pp. 3320–3328.
- You, S., D. Ding, K. R. Canini, J. Pfeifer, and M. R. Gupta (2017). Deep lattice networks and partial monotonic functions. In *NeurIPS*, pp. 2981–2989.
- Zambaldi, V., D. Raposo, et al. (2019). Deep reinforcement learning with relational inductive biases. In *ICLR*.
- Zhang, L., X. Li, M. Wang, and A. Tian (2021). Off-policy differentiable logic reinforcement learning. In *ECML/PKDD (2)*, Volume 12976 of *Lecture Notes in Computer Science*, pp. 617–632. Springer.
- Zhang, S., X. Ma, K. Duh, and B. V. Durme (2019). AMR parsing as sequence-to-graph transduction. In *ACL (1)*, pp. 80–94.

## BIOGRAPHICAL SKETCH

Harsha Kokel is a PhD Candidate in the department of Computer Science (CS) at The University of Texas at Dallas. She is advised by Professor Sriraam Natarajan. Harsha's research focuses on efficient knowledge-guided learning in structured, relational domains. She is also interested in sequential decision-making problems and, specifically, exploring the combination of planning and reinforcement learning. Her research has been published in the following conferences: AAAI, ICAPS, NeurIPS, and ACS. Her work has been presented at various workshops at AAAI, NeurIPS, IJCNN, ICAPS, and RLDM. She has reviewed papers for various journals, conferences, and workshops, including AAAI, IJCAI, SDM, and DMKD. She is a co-organizer for the PRL workshop at ICAPS 2023. She serves as an assistant electronic publishing editor for JAIR.

She received a master's degree in CS from The University of Texas at Dallas in 2021. She was a research intern at IBM T. J. Watson Research Center in the summer of 2021 and 2022. In the summer of 2018, she was a Machine Learning intern at Turvo Inc, CA. Before pursuing graduate school, Harsha provided content management solutions as a software engineer for four years; three years at Publicis Sapient Consulting and a year at Amadeus Software Labs. She received her bachelor's degree in Information and Communication Technology (ICT) from Dhirubhai Ambani Institute of ICT, Gandhinagar, India in 2012. During her bachelor's study, she was a research assistant with Prof. Prasenjit Majumder, working on Sandhan, a cross-lingual search engine for Indian languages. She co-organized a Morpheme Extraction Task at Forum for Information Retrieval and Extraction (FIRE) in 2012 and 2013. She also conducted a tutorial on Information Retrieval at Microsoft Research, India in 2013 as a part of Pre-FIRE Workshop.

## CURRICULUM VITAE

# Harsha Kokel

### Contact Information

Department of Computer Science  
The University of Texas at Dallas  
800 W. Campbell Rd.  
Richardson, TX 75080-3021, U.S.A.

Email: [hkokel@utdallas.edu](mailto:hkokel@utdallas.edu)  
Website: [www.harshakokel.com](http://www.harshakokel.com)

### Education

University of Texas at Dallas [Fall '18 - present]  
PhD, Computer Science,  
Advisor: Prof. Sriraam Natarajan  
Thesis: Beyond Data: Efficient Knowledge-guided Learning for Sparse and Structured Domains

University of Texas at Dallas [Fall '17 - Spring'21]  
MS, Computer Science (GPA: 3.961),  
Advisor: Prof. Sriraam Natarajan

Dhirubhai Ambani Institute of ICT (DA-IICT), Gandhinagar, India [May '13]  
BTech, Information and Communication Technology (GPA: 3.4),  
Advisor: Prof. Prasenjit Majumder  
Thesis: Language identification for short text in transliterated space

### Employment History

Research Assistant, UT Dallas, TX (Spring '19–Fall '22)  
Instructor, UT Dallas, TX (Fall '21)  
Research Intern, IBM T. J. Watson Research Center, NY (Summer '21 & '22)  
Teaching Assistant, UT Dallas, TX (Fall '18)  
ML Intern, Turvo Inc., CA (Summer '18)  
Senior Software Engineer, Amadeus Software Labs, Bangalore, India (2016–2017)  
Associate Technology, Publicis Sapient Consulting, Bangalore, India (2013–2016)  
Research Assistant, DA-IICT, Gandhinagar, India (2012–2013)

## Professional Service

Co-organizing a workshop on Bridging the Gap Between AI Planning and Reinforcement Learning (PRL) at ICAPS 2023.

Assistant Electronic Publishing Editors for JAIR (2020 - present).

Reviewer for journals, including Data Mining and Knowledge Discovery Journal and Big Data Journal.

PC Member for conferences, including AAAI-23 (main track and AI for social good track), IJCAI 2022 (Demo Track), CODS-COMAD 2020, and SDM 2020.

Reviewed papers for workshops, including ML Reproducibility Challenge 2021 Fall, Women in Machine Learning (WiML) 2021, Workshop on Graphs and more Complex structures for Learning and Reasoning (GCLR) 2022, and I Can't Believe It's Not Better (ICBINB) Workshop 2021.

Conducted a tutorial on Information Retrieval in Microsoft Research India & IRSI Pre-FIRE workshop, 2013.

Co-organized Morpheme Extraction Task at FIRE 2012–2013.

## Publications

### Book Chapters

**Kokel, H.**, Ramanan, N., Odom, P., Blasch, E., and Natarajan, S., *Human-Allied Learning of Probabilistic Models from Relational Data*, Chapter in *Handbook on Dynamic Data Drive Application Systems (DDDAS) (Vol. II) (to appear)*.

### Journal Papers

**Kokel, H.**, Balaraman, R., Tadepalli, P., and Natarajan, S., *RePReL: A Unified Framework for Integrating Relational Planning and Reinforcement Learning for Effective Abstraction in Discrete and Continuous Domains*, **Neural Computing and Applications 2022**, Special Issue on Human-aligned Reinforcement Learning for Autonomous Agents and Robots.

### Conference Papers

**Kokel, H.**, Das, M., Islam, R., Bonn, J., Cai, J., Dan, S., Narayan-Chen, A., Jayannavar, P., Doppa, J.R., Hockenmaier, J., Natarajan, S., Palmer, M., and Roth, D., *LARA – Human-guided collaborative problem solver: Effective integration of learning, reasoning and communication*, In **Advances in Cognitive Systems (ACS) 2022**.

Karanam, A.\*, Killamsetty, K.\*, **Kokel, H.\***, and Iyer, R. *Orient: Submodular Mutual Information Measures for Data Subset Selection under Distribution Shift*, In **NeurIPS 2022** (\*Joint first authors).

**Kokel, H.**, Prabhakar, N., Balaraman, R., Blasch, E., Tadepalli, P., and Natarajan, S., *Hybrid Deep RePreL: Integrating Relational Planning and Reinforcement Learning for Information Fusion*, In **FUSION 2022**.

**Kokel, H.**, Das, M., Islam, R., Bonn, J., Cai, J., Dan, S., Narayan-Chen, A., Jayannavar, P., Doppa, J.R., Hockenmaier, J., Natarajan, S., Palmer, M., and Roth, D., *Human-guided Collaborative Problem Solving: A Natural Language based Framework*, In **ICAPS (demo track) 2021**.

Karanam, A., Hayes, A., **Kokel, H.**, Haas, D., Radivojac, P., and Natarajan, S., *A Probabilistic Approach to Extract Qualitative Knowledge for Early Prediction of Gestational Diabetes*, In **AIME 2021**.

**Kokel, H.**, Manoharan, A., Natarajan, S., Balaraman, R., and Tadepalli, P., *RePreL : Integrating Relational Planning and Reinforcement Learning for Effective Abstraction*, In **ICAPS 2021**.

**Kokel, H.**, Odom, P., Yang, S., and Natarajan, S., *Unified Framework for Knowledge Intensive Gradient Boosting: Leveraging Human Experts for Noisy Sparse Domains*, In **AAAI 2020**.

Sankepally, R., **Kokel, H.**, Agarwal, K., and Majumder, P., *Morpheme Extraction Task at FIRE 2012-2013*, In Post-Proceedings of **FIRE 2012 and 2013**, ACM

## Workshop Papers

**Kokel, H.**, Lee, J., Katz, M., Sohrabi, S., and Srinivas, K. *Action Space Reduction for Planning Domains*, **Planning and RL (PRL) Workshop at ICAPS 2022**.

**Kokel, H.**, Manoharan, A., Natarajan, S., Balaraman, R., and Tadepalli, P., *Deep RePreL-Combining Planning and Deep RL for acting in relational domains*, In **Deep RL Workshop at NeurIPS 2021**.

**Kokel, H.**, Manoharan, A., Natarajan, S., Balaraman, R., and Tadepalli, P., *Dynamic probabilistic logic models for effective abstractions in RL*, In **Statistical Relational AI (StarAI) Workshop at IJCLR 2021**.

**Kokel, H.**, Manoharan, A., Natarajan, S., Balaraman, R., and Tadepalli, P., *RePreL: Integrating Relational Planning and Reinforcement Learning for Effective Abstraction*, In **Planning and RL (PRL) Workshop at ICAPS 2021**. \*(Contributed talk, 11/25 accepted paper)

Wan, G. and **Kokel, H.**, *Graph Sparsification via Meta-Learning*, In **Deep Learning for Graphs (DLG) Workshop at AAAI 2021**. \*Contributed talk (4/22 accepted paper)

## Abstracts

**Kokel, H.**, Natarajan, S., Balaraman, R., and Tadepalli, P., *Dynamic probabilistic logic models for effective task-specific abstractions in RL (Abstract)*, In **RLDM 2022**.

**Kokel, H.**, Lee, J., Katz, M., Sohrabi, S., and Srinivas, K. *How to Reduce Action Space for Planning Domains?*, As **Student Abstract** in **AAAI 2022**. \*Oral presentation (19/111 accepted abstracts)