HUMAN-ALLIED EFFICIENT AND EFFECTIVE LEARNING IN NOISY DOMAINS

by

Mayukh Das

APPROVED BY SUPERVISORY COMMITTEE:

Sriraam Natarajan, Chair

Vibhav Gogate

Gopal Gupta

Dan Roth

This dissertation is dedicated

to my father who is not with us anymore,

but I do hope that I inherit his dreams and his wide eyes,

that used to marvel at every sight they saw,

and wonder - 'What a wonderful world!'

HUMAN-ALLIED EFFICIENT AND EFFECTIVE LEARNING IN NOISY DOMAINS

by

MAYUKH DAS, B.Tech, 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

December 2019

ACKNOWLEDGMENTS

I wish to convey a very special thanks to my advisor, Dr. Sriraam Natarajan, for his unending support, guidance and for teaching me, essentially, all I know about artificial intelligence as well as research in general. He has always motivated me to give the utmost value to scholarship. I intend to adhere to these principles in all my subsequent endeavors, professional or otherwise.

I would like to thank all of my advisory committee members, Dr. Sriraam Natarajan, Dr. Vibhav Gogate, Dr. Gopal Gupta and Dr. Dan Roth, without whose help, support and guidance this dissertation would not have been possible. I would like to thank Dr. Gautam Kunapuli; from the time he became a part of our research group, I have learnt a lot from him. I have had the opportunity to collaborate with him on multiple papers. His guidance has been one of the major contributors to their acceptance and publication in important journals.

Trying to acknowledge my parents and my wife would amount to undermining their role in my life, both personal and professional. Whatever I have managed to become today, is largely their contribution. A few pages is not enough to describe the hardships my parents have gone through, just to see me progress in my career. The same goes for my wife, who has been my conscience since the day we got together. She transformed me as a person, from being a shy, introvert and opinionated human being, to a person who realizes the value of positive interactions all too well. She has survived alone, being far apart, and yet stood by me every step of the way.

I would like acknowledge my peers in the StARLinG Lab (at UTD), who have not only been my colleagues but my closest friends as well. Surviving my graduate life with my sanity intact was essentially possible because of their presence in my life, especially during the tragic events that transpired during these years. Aside from those, we have had the opportunity to have very motivating research discussions and to collaborate on several research ventures.

I would like to specially acknowledge some of my closest friends, outside my workplace, who have been with me all my life, through successes and failures. They, supported me when I was fighting

through difficult times and pushed me when I was complacent. They also punched me back to my senses when I was anxious, as well as showered me with their affection when I needed it.

I would also like to acknowledge Dr. Dan Roth (University of Pennsylvania). I have had the opportunity to work and collaborate with him as part of the DARPA Communicating with Computers (CwC) project. His wisdom has helped me have a clearer understanding of some of the problems I have tried to address in this dissertation. In fact, some of the terms used in this dissertation were coined during our discussions within the CwC group. I would also like to thank Dr. Janardhan Rao Doppa (Washington State University), with whom I have collaborated and co-authored multiple peer-reviewed papers. His guidance, and attention to detail was an immense positive motivating factor in much of our research work in the context of Human-Allied AI (HAAI), outlined in this dissertation. I am also extremely grateful to Dr. Kristian Kersting (TU Darmstadt, Germany), with whom I have had the opportunity of collaborating on several occasions. His vision, wisdom and guidance helped me expand my perspectives and made the research work stronger, resulting in acceptance at well-known conferences. I would also like to thank Dr. Yuqing Melanie Wu (Pomona College, Claremont, CA; previously at Indiana University, Bloomington (IUB)), who has been my inspiration early on in my graduate life. I thank her for introducing me to my advisor.

I would like to thank all the staff of the Computer Science Department at The University of Texas at Dallas for helping me complete administrative process in the smoothest fashion, be it for travel, for event organization or for academic formalities. A special mention is necessary for the staff and professors at IUB as well as the city of Bloomington, IN, where I spent half of my graduate life. IUB will always hold a special place in my heart.

Finally, I gratefully acknowledge the support of DARPA CwC Program Contract W911NF-15-1-0461. Any opinions, findings and conclusions expressed in this dissertation are those of the author(s) and do not necessarily reflect the view of the DARPA, ARO or the US government.

October 2019

HUMAN-ALLIED EFFICIENT AND EFFECTIVE LEARNING IN NOISY DOMAINS

Mayukh Das, PhD
The University of Texas at Dallas, 2019

Supervising Professor: Sriraam Natarajan, Chair

AI agents must work in alliance with the humans, be it in mundane activities or in critical and strategic tasks to realize their full potential. This is crucial to envisage the objective of developing intelligent agents that can learn an act in noisy structured environment and are robust to observation density and quality. While extensive research and sophisticated techniques exist for faithfully modeling complex and rich information in structured noisy domains, robustness to density and quality are open challenges.

Though AI agents are expected to learn/reason better with more data, yet arbitrarily large/dense data sets may, instead, lead to inefficiency, especially in structured domains. The problem is even more critical in sequential decision making, where the agent has to learn or reason continuously. The challenge of observation-quality, though distinct, is not completely orthogonal to the problem of density. Low-quality observations may originate due to systematic signal error, cognitive bias in data persistence as well as sample sparsity (known as systematic noise in general). Thus the two challenges are closely coupled. When the density becomes extremely low, it affects quality.

Naive solutions for such interconnected and closely-coupled challenges, such as powerful computational hardware or explicit noise modeling with the help of gold-standard observations can be insufficient. This is due to the fact that the real problem lies at a deeper, more conceptual level and there is a need to develop domain independent solutions. For instance, it is difficult to model sys-

tematic noise, even reasonably well, using most existing noise modeling approaches and presence of gold-standard observations is a strong assumption.

This dissertation investigates these challenges in depth and presents our unified Human-Allied AI (HAAI) framework to address them. It highlights some crucial insights in the context of both the challenges we address. Detailed algorithms are presented and analyzed as needed. We gain clarity about how Human-AI alliance, while not the more popular competitive or adversarial setting, is still a reasonable approach to address the above challenges. We also understand what a true HAAI system should entail and this work is a significant step in that direction. For instance progressive knowledge enhancement of the AI agent is one of the key aspects in HAAI, but is extremely difficult in real-time while the agent interacts with a human. One of subsequent chapters proposes principled approach to achieve the same. In summary this dissertation takes a significant step towards developing a robust intelligent agent. Finally, it outlines the open problems that are integral in developing true *AI with common sense*.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Inspired by John McCarthy's grand vision (McCarthy, 1968) of AI with common sense, our objective is to develop intelligent agents that can efficiently and effectively reason, make decisions and act in noisy complex and structured environments. To realize the true potential of Artificial Intelligence, it is essential that AI agents are perceived as allies to humans, facilitating in tasks ranging from mundane perception tasks, such as sensor/device control, to higher-level congnitive and strategic tasks, such as decision modules in deep space rovers, market intelligence systems, clinical decision support systems etc. Such AI agents should be robust to both quality and density of observations[1].

Given: Complex environment with *implicit domain structure* and potentially noisy *noisy* observed data.

To Do: Develop an AI agent that learns to act in such environments, using varied forms of observations, and be robust to quality and density of observations.

Example 1. *Consider a clinical decision support system that facilitates a physician to create suitable treatment plans for patients or the hospital administration in making strategic decisions about health-care practices (Fig. 1.1). Such a system may have access to different kinds of observations (data) such as Electronic Health Records (EHRs) of patients, drug reactivity databases, imaging data and so on. Such observations are at varied levels of representation; from low-level signals to structured databases containing entities, relations and hierarchies. Also, they could be noisy due to possible errors in observation, data-entry or implicit lack of samples.*

The AI agent must learn and reason with such varied forms of observationsFor effective treatment planning, it should predict potential adverse effects of drugs for diagnosis or track patient

[1]We use 'observations' and 'data' interchangeably throughout.

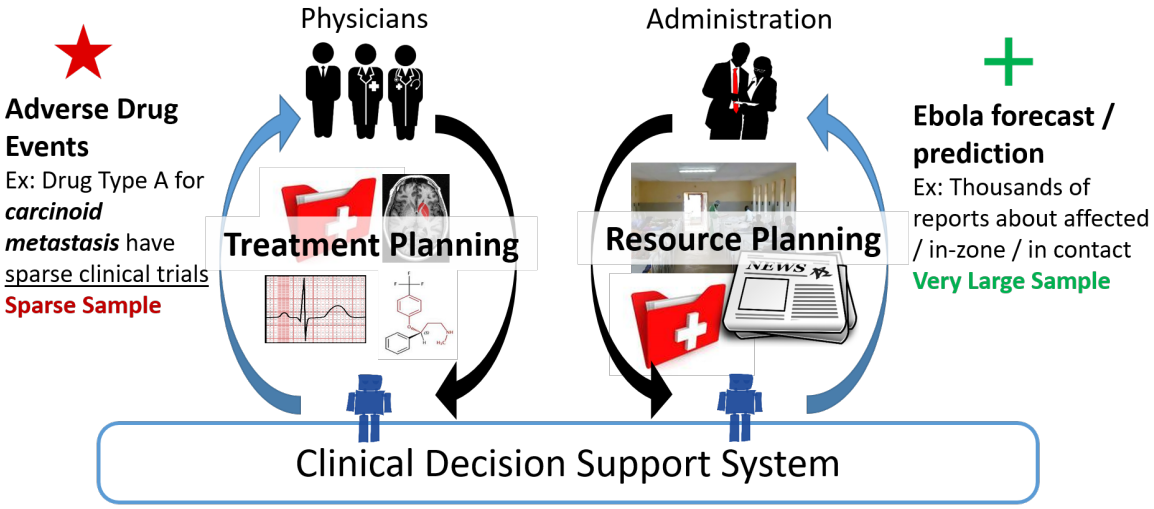


Figure 1.1: Our envisioned intelligent clinical decision support agent

state/condition at it evolves, for suggesting decisions in a treatment plan. It should also be able to learn/reason about various clinical and financial resources for providing optimal healthcare.

Real environments are complex and exhibit inherent structures such as objects, relationships and hierarchies, irrespective of how we encode them. An AI system must be able to represent and reason with such underlying structures to appropriately characterize the environment complexity. A flattened vectorized representation of such complex interactions, for traditional machine learning, is not sufficient since it results in loss of critical information (Jensen and Neville, 2002). Extensive research provides us with the sophisticated machinery which can faithfully model both implicit structure and noise, thereby allowing for learning and acting in such environments. Statistical Relational Learning (SRL/StarAI) (Getoor and Taskar, 2007; Raedt et al., 2016) combines a symbolic structured representation (such as First-Order Logic) with probabilistic semantics to faithfully model, learn and reason about in structured noisy domains both in the context of predictive modeling (Domingos and Richardson, 2004; Domingos and Lowd, 2009; Getoor et al., 2001; Kersting and De Raedt, 2007; Muggleton, 1996; Poole, 2003; Natarajan et al., 2012) as well as sequential decision making and acting, such as Reinforcement Learning, Planing/Probabilistic-Planning, Imitation Learning etc. (Tadepalli et al., 2004; Ratliff et al., 2009; Lang and Toussaint,

2010; Natarajan et al., 2011). However, there are two key challenges we need to address for realizing our envisioned robust AI agent.

1. **Observation density**: Class of approaches that enable effective learning/reasoning have a major limitation; they are difficult to scale to arbitrarily large samples (data-dense)

2. **Observation quality**: Effective, learning or acting with *sparse*, or in general *low quality*, samples is a provably difficult task. It is even more critical in structured environments where the state/feature spaces are intractably large.

1.1 Observation Density - Scalablility

While SRL has been effectively and successfully adapted for many complex problems including recommendation systems (Yang et al., 2017) etc, scaling learning/inference in SRL to arbitrarily large data sets (dense observations) remains a challenge. Substantial research exists on efficient learning/inference in SRL (Natarajan et al., 2012; Khot et al., 2011; Schapire and Freund, 2012; Singla and Domingos, 2008; Poole, 2003; Niepert, 2012; Kersting et al., 2009). However, most SRL formulations involve a fundamental, albeit hard (#P-complete), operation – counting satisfied instances of logic rules. It signifies computing the cardinality of the set of supporting instances, also known as coverage, of generalized compact features (in learning) or membership cardinality of equivalence class of indistinguishable instances (in lifted inference). Counting operation can become intractable as the density of observations increase, making both learning and inference inefficient and, thus, non-scalable with arbitrarily large samples.

Example 2. *The clinical decision support agent in Example 1 aims to facilitate resource planning for hospital administration. To fulfill that objective, it has to learn and predict/forecast potential disease outbreaks (such as Ebola). Such tasks require counting over large databases of patient records, news reports and related entities such as, contact with patients diagnosed with ebola or*

with persons from countries with reported Ebola cases, located in vicinity of ports of entry and so on, in order to estimate likelihood of an outbreak. Some example logic rules (model) could be,

$$\texttt{Prep}(\texttt{h},''\texttt{Ebola}'') : -\texttt{Nb}(\texttt{h},\texttt{n}), \texttt{In}(\texttt{n},\texttt{p}_1), \texttt{Contact}(\texttt{p}_1,\texttt{p}_2), \texttt{Confirmed}(\texttt{p}_2,''\texttt{Ebola}'') \tag{1.1}$$

$$\texttt{Prep}(\texttt{h},''\texttt{Ebola}'') : -\texttt{Nb}(\texttt{h},\texttt{n}), \texttt{In}(\texttt{n},\texttt{p}_1), \texttt{Contact}(\texttt{p}_1,\texttt{p}_2), \texttt{From}(\texttt{p}_2,\texttt{c}_1), \texttt{Reported}(\texttt{c}_1,\text{“}\texttt{Ebola}'') \tag{1.2}$$

where `Prep(h)` is a predicate that a hospital h should prepare for `''Ebola''` outbreak. `Nb(h,n)` signifies the neighborhood of h, `In(n,`p_1`)` signifies if person p_1 lives in neighborhood n and has ever had any contact with person p_2 (`Contact()`) with confirmed case of Ebola. Also `From(`p_2, c_1`)` denotes that person p_2 is from a country c_1, which has reported cases of Ebola.

To fully appreciate the complexity of the problem, consider that there could be millions of people living in the neighborhood, who could have had contact with thousands of incoming travelers from 100s of other countries. Computing the coverage (satisfied instances) of these rules is a challenging task, even more so due to the structured nature of the domain with entities (hospitals, people, countries, neighborhoods etc.) and interactions among them. Also, the patients already admitted for treatment of some other condition can all be clustered into a group where they are indistinguishable form each other since they are not of any interest in the given task. Size/count of such groups are also required to efficiently infer, about the set of people not relevant to Ebola outbreak, by exploiting symmetries. With large data sets, such counting operations are intractable. However, (explained in later sections) exact values of counts are inconsequential in large samples especially in classification tasks — 1000 is no worse then 992 persons, who satisfy the rules, when predicting an Ebola outbreak. **We develop fast approximation strategies for counting either satisfied instances given the constraints/logic rules or cluster sizes of mutually indistinguishable entities given the task.** Thus, we enable efficient learning/inference with arbitrarily large data sets (observation-dense environments).

Given: Arbitrarily large samples (dense observations) in structured noisy domains

To Do: Efficient/scalable learning and lifted inference via approximate counting.

1.2 Observation Quality

Another challenge is that of learning with low quality observations. By low quality observations we refer to presence of *systematic noise* in the data. Systematic noise (Barber et al., 2009; Krippendorff, 1970; Gove et al., 1976) is different from random noise and can generally be attributed to the following sources.

- **Sparse observations/samples:** Sparse samples (low density observations) fail to effectively characterize important regions of the feature space, unless fair sampling is ensured *For instance: lack of sufficient clinical trials for a new drug*.

- **Cognitive bias:** Data in the real-world is the outcome of various activities and transactions recorded regularly. Such data is prone to our social, geographical, ethnic or other biases *For instance: Ethnic and social biases in mental health treatment data (Snowden, 2003)*

- **Observation error:** Error in subsets of observed attributes is distinct from random signal noise. *For instance: In a smart home for an elderly patient, a subset of the sensors may lose calibration and send erroneous signals with respect to a particular location in the house.*

Note that, impact of observation quality is not orthogonal to observation density. Sparse observations, as discussed earlier, is one of the sources of systematic noise (*i.e.* low quality data). Systematic noise leads to generalization error (Niyogi and Girosi, 1996). Unlike random noise, it is difficult to detect or characterize systematic noise without additional knowledge or presence of supporting error-free observations. Thus, effective learning with systematic noise is challenging. The principle of structural risk minimization (Vapnik, 1999) also shows how optimal generalization from extremely sparse observations is quite hard.

Knowledge-based approaches have been proven to be widely successful in effective learning with sparsity (*i.e.* data scarce environments), or systematic noise in general (Towell and Shavlik, 1994; Fung et al., 2002; Odom and Natarajan, 2018). Mitchell (1980) postulated that inductive

bias is essential for true generalization over new instances. Augmenting learning with knowledge from domain experts provides such inductive bias. This inspires our objective of human-in-the-loop collaborative learning and problem solving framework to tackle the challenge of observation quality. We term this phenomenon as **Asymmetry of knowledge**[2] where humans and intelligent agents have complementary set of competencies (not necessarily mutually exclusive).

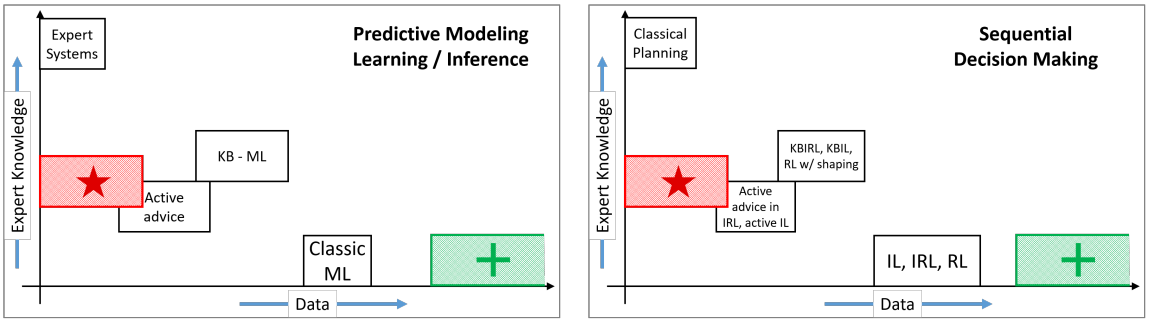
Example 3. *The AI agent in Figure 1.1, may have powerful reasoning engines, access to patient's medical records and so on. However, for the drug groups meant for carcinoid metastasis, there are insufficient clinical trials making it difficult for the agent to effectively learn/predict about any adverse side effect (ADE). If the agent can collaborate with a biomedical expert and leverage additional knowledge about some biochemical properties of the drug group that the physician aims to prescribe, then the agent could possibly to draw better conclusions.*

In this age of big data, most predictive/analytics/decision-making systems tend to be completely data-centric and we fail to consider that data in the real world is sparse, incomplete, noisy, unbalanced and mostly non-representative of the true concept. The problem of systematic noise and sparsity is more critical in structured/relational spaces since most relations are false in the world. While existing knowledge-based approaches have allowed for enhanced model learning and decision-making, they have limitations including problem-specific representation, the way human experts are leveraged or even the class of problems considered therein. **In this work, our objective is to develop Human-Allied AI framework that allows the an AI agent be robust to systematic noise, especially due to sparse samples (data-scarce environments).**

Given: Structured noisy environments with low-quality observations (sparse samples, cognitive biases) and access to human expert(s)

To Do: Human-Allied AI for effective learning and sequential decision making

[2]The term was coined from discussions within the DARPA Communicating with Computers (CwC) research group



(a) Predictive Modeling (b) Sequential Decision Making

Figure 1.2: Our problem space with respect to Learning/Inference in predictive modeling *(left)* and Sequential Decision Making *(right)*. Red with asterisk is the part of the problem space with sparse observations and the Green with '+' is the part where observations are dense requiring scalable approaches. This dissertation investigates both.

1.3 Dissertation Statement

This work aims to investigate the following: *Approaches for effective and efficient learning/reasoning in structured domains, robust to the challenges of both observation quality and density (sparse as well as arbitrarily large samples), in the context of predictive modeling, sequential decision-making or problem solving, across different levels of data representation.*

1.4 Dissertation Contributions

This dissertation makes the following contributions towards the envisioned HAAI framework:

I. ***Scaling relational learning/reasoning to large data, via effective count approximation***

II. ***Effective learning in structured domains with sparse noisy samples (low quality observations) via human-AI collaboration***.

(i) **Collaborative problem-solving** (planning) in structured spaces but no demonstrations

(ii) Guided learning in **low-level representations**

(iii) Human-guided **learning of novel concepts** from sparse instances.

1.5 Dissertation Outline

The dissertation has been divided into two high-level parts. Part I outlines our approaches for efficient learning and inference with arbitrarily large data and Part II outlines the Human-AI collaborative frameworks for learning and decision-making in data-scarce environments.

Chapter 2 presents the necessary background and related work and positions our work in the context of current research. First we discuss some concepts and existing research on learning and decision-making in structured noisy domains, which lays the groundwork for this dissertation. Subsequently, we outline the existing perspectives towards scaling and efficiency in reasoning with structured data. We also present related literature on human knowledge augmented learning and sequential decision making both with structured as well as low-level representations.

Part I

Chapter 3 details our first proposed approach, *FACT*, for scaling learning and inference in structured noisy domains to arbitrarily large data sets via graph databases. It presents our novel count approximation framework that uses a message passing scheme to estimate frequencies using RDF graphs, followed by an extensive evaluation on 3 major tasks of modeling structured data - (a) Parameter Learning (b) Structure Learning and (c) Lifted Inference. We also discuss some adaptations of our approximation technique for efficient reasoning in several real-world tasks.

Chapter 4 presents a generalized extension (*MACH*) of our count approximation strategy. It addresses the limitations of our previous approach and other related approaches and proposes an approximate satisfiability-counting technique based on **hypergraph** summaries. Specifically it provides a principled formulation for efficiently estimating expected counts of satisfied instances of First Order Logic clauses via intelligent factorization of the distribution induced by a clause.

Part II

Chapter 5 presents, K-CLN, a human-guided training approach of structure-aware deep models for effective learning with sparse data. Deep models have recently gained significant attention in learning from low-level signals. Column Networks (CLN), a deep architecture, can also succinctly capture domain structure along with modeling low-level representations. But, they may still be prone to sub-optimal learning from sparse and noisy samples. Hence, we propose Knowledge-augmented CLN that leverage human advice/knowledge for noisy/sparse samples.

Chapter 6 presents PGPLANNER, an active preference elicitation framework for automated planning in AI. Planning, one of the several sequential decision making paradigms, uses a limited amount of domain axioms and performs intelligent search to infer a sequence of actions to achieve given goals. Unlike other paradigms, there is no data/demonstrations to learn from in automated planning. This chapter discusses and exhaustively evaluates this novel framework, where a planning agent can actively seek human guidance for finding better plans.

Chapter 7 investigates and proposes our concept learning framework that allows for induction of novel concepts from sparse data/demonstrations. Our GOCI framework, extends Inductive Logic Programming a novel conceptual distance metric and active human guidance to learn an optimally generalized representation of novel concepts from single (or few) instance(s). We provide empirical validation of its effectiveness as well as theoretical guarantees on both the validity of our novel metric and the PAC learnability in our problem setting.

Chapter 8 presents some additional challenges that I have explored as part of my dissertation, where we have either adapted some of the techniques outlined above or have proposed novel solutions. **Chapter 9** presents our concluding remarks and insights about our work discussed in the earlier chapters and finally introduces several open challenges that remain to be addressed to achieve a true human-AI collaboration and discusses their potential impact.

CHAPTER 2

TECHNICAL BACKGROUND

This chapter provides a comprehensive background on the problem space that this dissertation aims to address. We first describe that basic concepts that are essential in understanding the challenges involved in learning and acting in structured noisy domains. Next we discuss the extensive literature on modeling and learning/decision-making in structured spaces, their limitations and how they connect with the key contributions of this dissertation. Finally, we present and analyze the existing research that closely relate to the solution approaches we propose in the subsequent chapters.

2.1 Learning and Decision Making in Structured Noisy Domains

Observations/data acquired from real-world scenarios are implicitly structured. Most environments can be characterized in terms of entities and entity classes, their properties, relationships among them and their hierarchies. In fact, that is a natural cognitive strategy of humans. *For instance, in a treatment planning context (Example 1), to reason about the potential adverse effects of prescribed drugs it is necessary to model complex biochemical interactions when a drug is metabolized in a human body. Such interactions involve different entities including enzymes, antibodies and different compounds in the drug. Interactions of chemical nature among these entities or groups (hierarchies) of such entities constitute the structural information essential for prediction and discovery of adverse effects.* Thus intuitively, every environment can be modeled as a multi-relational, hierarchical, directed (or bidirected) graph.

Our representational choices merely affect our ability to preserve that rich structural information. A vectorized representation of observations is insufficient. *For instance, on one hand we cannot pre-determine the number interactions a drug is involved in with other biochemical compounds making it impossible to encode them as fixed length feature vectors and on the other hand feature vectors with respect of each drug makes it difficult to generalize over it's reactions with*

other drugs. Thus, we encounter two primary problems with vectorizing structured domains - 'linkage' and 'autocorrelation' (Jensen and Neville, 2002). This problem is equivalent to the problem of arbitrary number of (self)joins required in an SQL query to construct the correct view of the data. One may argue that some data could be implicitly unstructured such as text or image data. However, using a richer model can allow for inferring richer domain relationships. Our hospital resource planning scenario involves learning from news reports and broadcasts. However, there does exist implicit relationships such as entities (countries, patients, hospitals etc.). There could also be 'reference' relationships between news reports. Treating news reports as 'bag-of-words' unstructured data is, hence, a representational choice leading to loss of such vital information.

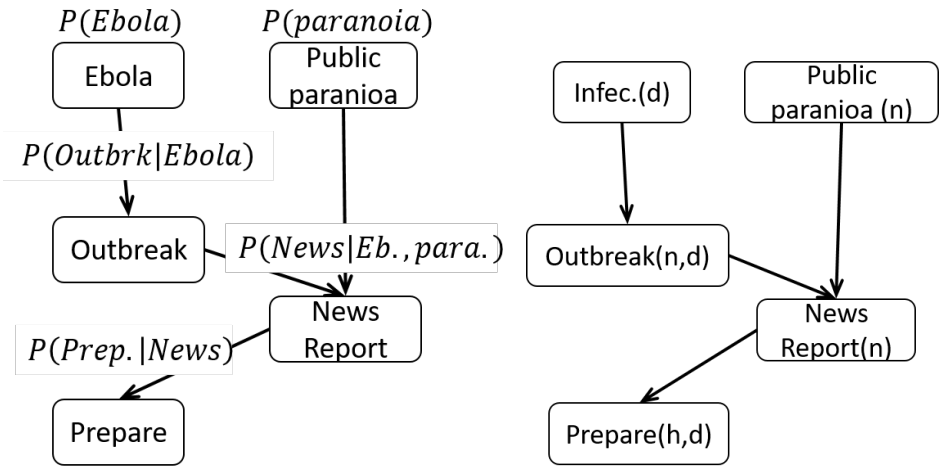
Extensive research on relational databases (Silberschatz et al., 1997) and graph databases (Angles and Gutierrez, 2008) allows us the privilege of structured representation in terms of persistent storage. It is necessary that predictive modeling or sequential decision making approaches exploit such rich structural information. Thus we need a representational language that is powerful enough to seamlessly capture relationships and hierarchies as well as generalize over classes and arbitrary-sized groups of entities. Logic (esp. First Order Logic, abv. FOL (Flach, 1994)) is an ideal solution since it is not just a powerful representational language but is also quite intuitive. For instance, our hospital resource forecasting model may have a rule such as,

$$\forall h, d, n, p_1, p_2\, \texttt{Prep}(\texttt{h}, '' \texttt{Ebola}'') : -\texttt{Nb}(\texttt{h}, \texttt{n}), \texttt{In}(\texttt{n}, \texttt{p}_1), \texttt{Contact}(\texttt{p}_1, \texttt{p}_2), \texttt{Confirmed}(\texttt{p}_2, '' \texttt{Ebola}'')$$

which signifies that any hospital h in any neighborhood n where any person p living in that neighborhood has come in contact with a confirmed case of an infectious disease such as Ebola, must start preparing for the outbreak of d (*i.e.* $\texttt{Prep}(\texttt{h}, \texttt{d})$). Note how important relations and attributes are captured via predicates such as `Contact()`, `In()`, `Confirmed()` etc. Most important aspect is the use of quantifiers $\forall$ and $\exists$ in First Order Logic (FOL), which allows required generalization and compact expressions. For instance, $\forall h, n, p_1, p_2$ generalizes over particular groups of hospitals in vicinity of people in contact with confirmed Ebola affected cases. Clearly, it is

intractable to capture patterns and rules about every possible combination of hospitals, people or neighborhoods. *Generalization is the key.* Expert systems such as 'Mycin' (Shortliffe, 1974a) encoded expert knowledge into such rules and reasoned with them for treatment planning. We go beyond expert systems and aim to learn from observations.

Inductive Logic Programming, ILP (Muggleton, 1991; Muggleton and De Raedt, 1994; Quinlan, 1990; Muggleton, 1995; Sammut and Banerji, 1986; Rouveirol, 1992, 1990; Srinivasan, 2007) inductively learns a logical program (first-order theory) that aims to cover most of the positive examples and none of the negative examples. Due to the representational power of First Order Logic, through the decades, this has proved to be an extremely effective modeling paradigm to faithfully capture and generalize structured information. With ILP, the goal is to generalize over instances using background knowledge as search bias by building valid hypotheses about unseen examples. While ILP allows for learning generalized compact features in structured spaces, real environments are noisy and uncertain as well. *For example, news articles have noise in them, such as they may have the word "Ebola" but are not actually reporting a confirmed cases or in countries with extreme paranoia, news reports are likely to inflate the amount of reported cases and so on. We also have to factor into our model, how rare a particular disease is.*



(a) Bayes Net (b) Parameterized BN

Figure 2.1: A Bayesian Network and its parameterized form for hospital resource forecasting.

Probabilistic models (Koller and Friedman, 2009), such as Bayes Nets, Markov Random Fields, Dependency Networks, provide effective techniques to model stochastic dependencies among random variables. However, such models are 'propositional'; *i.e.* they represent dependencies among features of a single entity and do not generalize over classes or groups or relationships among the same. *For instance, the Bayes Network in Figure 2.1(a) models the distribution over the outbreak of only one disease, "Ebola", and preparation of one hospital in one place.* It fails to generalize to other fatal infectious diseases and hospitals or important interactions such as vicinity and contact with other reported cases.

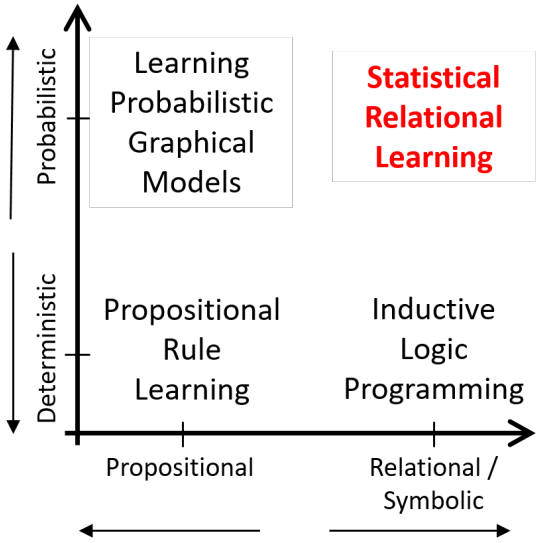


Figure 2.2: The space of SRL approaches

Statistical Relational Learning models (SRL) (Getoor and Taskar, 2007; Raedt et al., 2016) combine the representational power of FOL to capture rich structural information and the power of probability theory (statistical models in general) to model noise and uncertainty. An alternate perspective of SRL is of 'lifting' probabilistic graphical models, where probabilistic dependencies are captured among parameterized random variables (Neville and Jensen, 2007; Getoor et al., 2001). Parameterization allows for generalization over multiple objects. *For instance, Figure 2.1(b) illustrates the parameteried form of the outbreak model described earlier. Note that each node, now is*

paramaterized with a one or more logical variables, that allows for generalizing over all diseases that are infectious as well as all hospitals and neighborhoods and the dependencies among them. **Thus, the 'lifted' form of such models are essentially *templates* which will manifest a typical propositional graphical model if the logical variables are substituted with values.** How the conditional distributions are represented in such 'lifted' probabilistic models is beyond the scope of this dissertation [refer to Bayesian Logic Programs (BLPs) (Kersting and De Raedt, 2007) or Template Networks, such as Relational BNs (Jaeger, 2007)]. Clearly, both the representational perspectives of SRL are conceptually equivalent. For instance, the lifted BN discussed above can essentially be encoded as definite clause logic programs with contextual conditional distributions (as in BLPs). Henceforth, for clarity in explanation of various concepts in the rest of the dissertation, we will use the notion of combining FOL with probabilities as an unified view of SRL. Figure 2.2 illustrates how SRL is situated in the in the space of existing research. SRL approaches have generally ranged from directed models (Kersting and De Raedt, 2007; Koller, 1999; Heckerman et al., 2007; Kazemi et al., 2014; Neville and Jensen, 2007; Kazemi and Poole, 2018) to undirected models (Richardson and Domingos, 2006; Taskar et al., 2007; Kimmig et al., 2012; Kazemi et al., 2014; Fatemi et al., 2016). Any SRL model has 2 major components, structure and parameter. Structure refers to the the generalized compact patterns (relational features) represented via FOL formulas or through template graphs and parameters refer to quantities that indicate the relative importance of FOL rules or templates in the model. Such parameters can either be valid probability values or such unbounded weights that has to be normalized later to a valid distribution. Parameters are tied/shared between all the instances of the same template or rule. Below is an example of such a model, a Markov Logic Network (MLN) based on the disease outbreak planning in hospitals (Richardson and Domingos, 2006). Note that the weights on the left indicate that the

Table 2.1: Example MLN based on Hospital Resource Forecasting domain.

C_1	1.8	$\forall h, d, n, p_1, p_2$ $\texttt{Prep}(\texttt{h}, \texttt{d}) : -\texttt{Nb}(\texttt{h}, \texttt{n}), \texttt{In}(\texttt{n}, \texttt{p}_1), \texttt{Contact}(\texttt{p}_1, \texttt{p}_2), \texttt{Confirmed}(\texttt{p}_2, \texttt{d})$
C_2	0.5	$\forall h, d, n, p_1, p_2, c_1$ $\texttt{Prep}(\texttt{h}, \texttt{d}) : -\texttt{Nb}(\texttt{h}, \texttt{n}), \texttt{In}(\texttt{n}, \texttt{p}_1), \texttt{Contact}(\texttt{p}_1, \texttt{p}_2), \texttt{From}(\texttt{p}_2, \texttt{c}_1), \texttt{Reported}(\texttt{c}_1, \texttt{d})$

first rule is more important than the second one. Unlike that of PROBLOG (De Raedt et al., 2007) which uses valid probability values as weights on formulas, the weights in an MLN are unbounded real numbers. They akin to *log Odds* ($\log P/(1-P)$) for a random variable with binary outcomes.

2.1.1 Challenges in SRL

While these models are effective in modeling structured spaces faithfully and succinctly, learning them is computationally intensive. This is due to the following aspects:

- Like ILP, learning occurs at multiple levels of abstraction, – object classes, groups/sub-groups of objects and relations and possibly at the individual instances of the objects.

- Learning these models has two (not necessarily distinct) stages, namely, structure search and parameter estimation. Structure search, similar to ILP, requires estimating the support (instances in the data) of candidate formulas as well as their statistical measures which in turn requires parameter estimation. Again parameter estimation requires performing inference with the candidate models. Figure 2.3 illustrates the stages and their hierarchy.

Performing the different stages separately is challenging. Recent work on boosted approaches for SRL (Natarajan et al., 2012; Schapire and Freund, 2012) enables concurrent structure search and parameter estimation (full model learning), allowing efficient and effective learning.

Inference is equally challenging since, in a naive setting, it would require us to instantiate all the logical variables in the rules of the model structure with all the entities in the domain and construct a large probabilistic graphical model (each ground literal becomes a random variable) using the rule weights as shared potentials and then perform probabilistic inference. This could easily become intractable in large domains. *For instance, the MLN shown in Table 2.1 will reduce to a massive Markov Network when instantiated with all combinations of people, hospitals, countries, and diseases in the Hospital Resource Planning domain*. Lifted inference (Poole, 2003; Kersting

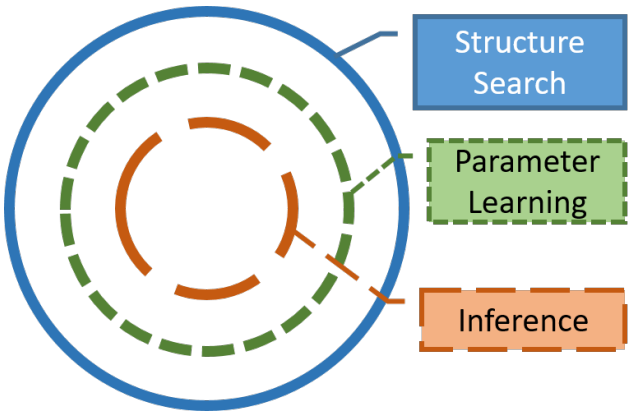


Figure 2.3: Multiple stages of learning Statistical Relational Models

et al., 2009; Singla and Domingos, 2008) alleviates this challenge to some extent by exploiting symmetries and inferring over groups of interchangeable objects as a whole.

Hence, there are two primary challenges in SRL, that aligns with the key challenges we are trying to address in this dissertation in order to develop our envisioned agent, are as follows:

- *Scaling to dense observations*: Both learning and inference in SRL are computational intensive tasks. While the approaches discussed above attempt to address this challenge to some extent, it is still an open problem. Several sampling based approaches (Venugopal and Gogate, 2014) and model reduction frameworks (Shavlik and Natarajan, 2009) attempt to address this as well by controlling the size of the resultant underlying ground model. However, there is a fundamental operation inside most SRL approaches - *counting* the number of satisfied instances of generalized compact patterns (r FOL rules) in the model structure. Counting is hard (#P-complete), making scaling learning and inference to arbitrarily large data sets (dense observations) extremely challenging. We explain in Section 2.2, the existing research that aims to address this challenge and how our approaches relate to them.

- *Sample sparsity:* At the other extreme is the challenge of sample sparsity. While systematic noise in general may exist irrespective of representational choices, sample sparsity is a critical challenge in structured domains. Structured domains may have arbitrary number

object classes, objects, relationship types among them. This makes the feature space exponentially large and even reasonable amount of observations is comparatively sparse and may not be optimally representative of the population distribution. *For example, in treatment planning context, for adverse effect detection, there could be millions of drugs interacting among each other as well as hundreds of metabolic chemical compounds, varied types of pre-existing physical conditions and so on. The feature space is in the order of millions of assertions and it is difficult to have a representative set of clinical observations.*

We present a more detailed background on the above challenges and related research in Sections 2.2 and 2.3.

2.1.2 Decision Making in Structured Noisy Environments

Sequential Decision Making in structured noisy domains require such generalized compact models as an internal reasoning engine. The high-level goal of sequential decision making is to learn (or reason with) a mapping from the state space to the action space. Such a mapping is either termed as a policy or, occasionally, an action model in planning ($\pi : \mathbb{S} \rightarrow \mathbb{A}$). Tabular encoding of this mapping via explicit state enumeration is only possible in domains with extremely small state spaces. To tackle this challenge a factored representation of the state space is used, all the way from early *search techniques* and planning to some of the more recent advances in RL (Sallans and Hinton, 2004). State spaces are factored when they are represented by a vector of features (properties of the domain). *For instance, in our treatment planning context a patient's clinical attributes, allergies, attributes obtained from current checkup etc., will be encoded as a feature vector. Changes in the values of any feature, say BP measure, will imply a change in state.* This allows us to use any machine learning technique of our choice to model state-action mapping. This is known as function approximation. However, as discussed earlier a vectorized representation is insufficient in characterizing structured, noisy environments. *For instance, it is challenging to capture information about clinical attributes of other patients who share kinship with the given patient in a vectorized*

representation of factored states. Similarly, it is impossible to generalize over arbitrary number of procedures and clinical tests or over other related entities such as compounds inside drugs. Thus SRL is indispensable for succinct factored representation (of policies/action models) for function approximation in structured noisy domains.

In the automated planning paradigm (Ghallab et al., 2004), domain definitions have long been encoded in restricted FOL representations, such as STRIPS and PDDL (Fox and Long, 2003), and planning problems have been posed as sequential logical inference problems. In probabilistic planning, the logical domain rules are softened with probabilities and interleaved logical inference and probabilistic inference is performed to generate plans. However, planning approaches based on Markov Decision Process (MDP) formulation as well as Reinforcement Learning (Sutton and Barto, 1998) paradigms are faced with the challenge of generalization and large state spaces. As is clear from the above example, factored MDPs and function approximation, partially address this challenge. However, vectorized representation of factored state spaces lead to the problem of loss of information (Jensen and Neville, 2002) as well as limits generalization across arbitrary objects. Relational Reinforcement Learning (Džeroski et al., 2001; Tadepalli et al., 2004) and related approaches address this by modeling the problem as a relational MDP and using statistical relational models as function approximators. Heirarchical environments structures have also been utilized in sequential decision making, such as MaxQ (Dietterich, 2000), semi-MDPs (Sutton et al., 1999), abstract-MDPs or hierarchical abstract machines (Parr and Russell, 1998) in reinforcement learning and Hierarchical Task Networks (Nau et al., 2003; Erol et al., 1994) in planning. Naturally, the challenges outlined in the context of SRL, directly apply to RRL and planning paradigms as well - scalability and sample efficiency. While scalabiltiy is essential for efficient decision-making, especially in high-impact domains such as treatment planning, the problem sparse observations/demonstrations is a more critical yet common problem. In large state spaces it is difficult to explore all potentially relevant regions. Also, some parts of the space may not even be safe to explore. *For example, it is impossible to obtain observations about fatal side effects of some drug*

compounds. Before we outline existing research on knowledge-based approaches that address the challenges of sample sparsity, we will briefly discuss automated planning, since we will present our Human-Allied AI frameworks in the context of planning.

Automated Planning Planning problems and automated planners are often characterized using state space models. However, the problems are presented to planners in the form of compact description in a suitable language (e.g., PDDL). A planning task can be seen as a 5-tuple $\langle S, A, I, G, T \rangle$ where: S is a (finite set of states; A is a finite set of actions, where $A(s) \subseteq A$ corresponds to actions that are applicable at state $s \in S$; $I \subseteq S$ is the set of possible initial states; $G \subseteq S$ is the set of possible goal states; $T : S \times A \mapsto S$ is the transition function, where $T(s, a)$ returns the next state for deterministic domains, and returns the probability distribution over the next states for stochastic domains. Additionally, in partially observable domains, some information of the state is hidden. Given an initial state and a goal specification as an instance of a planning task, automated planners produce a sequence of actions (aka. plan) to satisfy the goal specification. The most basic form of automated planning is computationally hard (specifically, PSPACE-complete (Bylander, 1991)). However, real-world applications require fast planners to satisfy time constrains. Consequently, there is a large body of work to address this challenge (Ghallab et al., 2004). Some representative approaches include reduction to SAT solving (Kautz and Selman, 1992, 1996; Blum and Furst, 1997), forward state space search with human-designed heuristics (Hoffmann and Nebel, 2001; Yoon et al., 2007), learned heuristics from solved planning problems (Yoon et al., 2008; Xu et al., 2009, 2010); planning with human-written control knowledge (Erol et al., 1994; Bacchus and Kabanza, 2000) and solving probabilistic planning via reduction to deterministic planning (Yoon et al., 2008). Hierarchical planning methods, such Hierarchical Task Networks (HTN) (Erol et al., 1994), decompose bigger problems into smaller ones recursively until it is relatively simple to obtain a partial plan for a sub-problem. Such technique closely resemble how humans intuitively reason about problems and make decisions.

2.1.3 Relation-aware deep model

Our envisioned AI agent (Example 1) must be able to learn and reason with varied levels of observations. *For example, the treatment planner should reason with low-level signals such as ECG signals, MRI scans etc. On the other side the resource forecasting AI agent has to reason with news articles, broadcasts and tweets etc.* Classical connectionist models (Rosenblatt, 1962), and more recent deep architectures (Goodfellow et al., 2016), have had considerable success in a variety of tasks involving such low-level signals. However, most deep models work with vectorized representation of the feature space resulting in information loss discussed earlier. Column networks transform relational structures into a deep architecture in a principled manner and are designed especially for collective classification tasks (Pham et al., 2017). Note that the *'Curse of dimensionality'* of the parameter space, results in the requirement for large amounts of data for effective training of such models. The problem is even more critical in structured spaces making effective trainign difficult with sparse samples (low quality observations). Hence, in Chapter 5 we present our knowledge augmented deep learning framework, that uses human advice for effective training with sparse samples. The architecture and formulation of the column network are suited for adapting it to the advice framework. The GraphSAGE algorithm (Hamilton et al., 2017) shares similarities with column networks since both architectures operate by aggregating neighborhood information but differs in the way the aggregation is performed. Graph convolutional networks (Kipf and Welling, 2016) is another architecture that is very similar to the way CLN operates, again differing in the aggregation method. Diligenti et al., (2017) presents a method of incorporating constraints, as a regularization term, which are first order logic statements with fuzzy semantics, in a neural model and can be extended to collective classification problems. Semantic-based regularization method presented in Diligenti et al., (2017) is similar in spirit to our proposed approach, but is differs in its representation and problem setup.

2.2 Scaling with dense observations

As discussed earlier, *counting* is a fundamental challenge in learning and acting in structured spaces, that needs our attention. Most SRL models (even ILP) require counting over the satisfied instances of FOL rules (relational features or generalized compact patterns) both for estimating support/coverage during structure search and for estimating probabilities during parameter estimation. For example, we may compute the likelihood of clauses to estimate support of a particular clause to be added to the model. For clause C_1 the likelihood could be[1],

$$L = \frac{\#[\texttt{Prep}(\texttt{h},\texttt{d}), \texttt{Nb}(\texttt{h},\texttt{n}), \texttt{In}(\texttt{n},\texttt{p}_1), \texttt{Contact}(\texttt{p}_1,\texttt{p}_2), \texttt{Confirmed}(\texttt{p}_2,\texttt{d})]}{\#[\texttt{Nb}(\texttt{h},\texttt{n}), \texttt{In}(\texttt{n},\texttt{p}_1), \texttt{Contact}(\texttt{p}_1,\texttt{p}_2), \texttt{Confirmed}(\texttt{p}_2,\texttt{d})]} \tag{2.1}$$

Similarly for computing distributions we require counts as well. For instance, for an MLN the joint distribution for a given world $X = x$ is expressed as $\frac{1}{Z}\exp\sum_{f\in F} w_f n_f$, where Z is the partition function (normalization constant) and F is the set of formulas in the model and n_f is the number of groundings or satisfiability count of a formula f. Thus for our MLN in Table 2.1, for a world/context of "Ebola" and "Parkland hospital" we get,

$$P(\theta) = \frac{1}{Z}\exp 1.8 \times \#[C_1/\theta] + 0.5 \times \#[C_2/\theta] \tag{2.2}$$

where, θ is the partial substitution (refer FOL.), $\theta = \{h/''Parkland'', d/''Ebola''\}$.

In lifted inference as well, counting is an essential operation to compute the membership cardinality of groups and sub-groups on which inference is performed. *For instance, in the MLN example in Table 2.1, if we are interested in inferring the probability of preparing 'Parkland' for* `Ebola` *then all the neighborhoods who are not in its vicinity or people in contact with patients of other infectious diseases are of no interest and hence will be treated symmetrically and clustered into one group.* However, we need the size of that group to compute the marginal over that group which will then be used during normalization. This problem is known as the satisfiability

[1]We use $\#[]$ to signify counts/cardinality of satisfied instances.

counting problem (#SAT) (Gu et al., 1996). It is a hard problem (#P-complete) and hence will render learning and inference non-scalable to large evidence/data. Chapters 3 & 4 presents our novel approaches for efficient approximation of this counting operation which allows us to scale SRL to arbitrarility large data.

2.2.1 Approximate subgraph matching

Venugopal et. al's work (2015) is closest in spirit, which scales via counting only the satisfied groundings in MLNs. We aim to keep our approach model independent. The key idea behind our work is that counting in SRL is akin to subgraph matching / enumeration, which when done in exact fashion is another $\#P$*-complete* (Valiant, 1979; Vadhan, 2001) problem in itself. Hence, as we show we Chapters 3 and 4, we perform approximate subgraph enumeration. Use of graphs in ILP and SRL is not new, as seen in Richards and Mooney's work on relational path finding (Richards and Mooney, 1992). However, they are restricted to exploiting the logical structure and do not perform fast or approximate counting as we do. Our work closely relates to *approximate* sub-graph counting, in a graph theoretic context (Slota and Madduri, 2013; Bhaskara et al., 2010). Most of such approaches focus, either on exploiting high performance architectures via parallelization, subject to resource availability, or on utilizing properties of restricted classes of graphs. *FACT* (Das et al., 2016), presented in Chapter 3 approximates counts via summarization of in-memory Property Graphs (Corby et al., 2000) encoded using RDFs [2] but suffer from certain fundamental limitations such as assuming binary relations and independence across relations sharing entities. ISMA (Demeyer et al., 2013) adopts a search tree optimization and pruning approach for subgraph enumeration in large biological networks. It has similar limitations and returns all subgraphs instead of count estimates. Fürer and Kasiviswanathan (2014) propose a polynomial time sampling-based approximation strategy for counting isomorphic subgraphs matching a given

[2] https://www.w3.org/RDF/

template leveraging bounded-width *ordered bipartite decompositions* of the templates. A key feature of this approach is its provable generalization across varied classes of graphs. Ravkic et al. (2018) extend this principle for parameter learning in SRL via efficient computation of the decompositions. While potentially applicable in our context, these approaches have major limitations that include restricted arity of relations and requirement of decomposability of templates.

A hypergraph representation alleviates some of these limitations as we show in Chapter 4. Approximate matching has been extensively studied in graph theory, recently in the context of hypergraphs (Dudek et al., 2014, 2013). While these approaches are interesting with theoretical guarantees, they apply to specific classes of simple hypergraphs with bounded degree and regularity which cannot model real-world multi-relational data. *For instance, in a treatment planning scenario there could be a relation called* `LabTest(person,hospital,test)`. *Now "person" and "hospital" can also be involved in another relation, say a technician works in that hospital,* $\texttt{WorksIn}(\texttt{person}, \texttt{hospital}) \wedge \texttt{Technician}(\texttt{person})$. Adapting the above mentioned hypergraph approximation approaches could only be possible if we partition the multi-relational graph by the relation types and model each partition separately. For example we could create two different hypergraphs one with hyperedges of type "`LabTest`" and the other with `WorksIn`. An entity, say `Eva`, will be present as nodes in both the hypergraphs. However, since entities (ex: `person`) may participate in multiple types of relationships, creating partitions that preserve information of shared relations is impossible.

2.2.2 Cardinality estimation in databases

Our approximate counting strategy depends on probabilistic estimates about the presence of (hyper)edges (predicates) between nodes (constants). In essence, we are meta-learning the statistical estimates about the entities and relationships of various types. We elaborate in the subsequent chapters how we obtain such estimates from summary statistics. This is closely related to cardinality/selectivity estimation and has been deeply studied in the context of relational

databases (Schiefer et al., 1998), be it using incremental statistics across tables and frequently executed queries such as histograms (Seputis, 2000), VC dimension based summaries (Riondato et al., 2011). Probabilistic database systems, such as *'Tuffy'* (Niu et al., 2011) exploit the power of relational databases for persistence and computations in SRL. While successful on standard benchmarks, these approaches are not directly applicable to our problem setting where we are interested in learning from adhoc databases. Also, such systems may encounter challenges of relational databases including arbitrary join costs and may require carefully designed join caches for optimization. More importantly, they have not yet been applied to the challenging task of full model learning in SRL (only parameter-learning). Graph-based systems can potentially alleviate such limitations while allowing seamless representation of logical assertions. However, cardinality approximation is an open problem in the context of Graph databases (Neumann and Moerkotte, 2011; Stocker et al., 2008), especially for hypergraphs. The complexity lies in the fact that graphs can potentially be multi-relational with the freedom of encoding infinitely many relations among infinitely many different types of entities. Thus there is no underlying "schema" to a graph-structured database. Hence, histogram based cardinality estimation methods that work well with a relational database and SQL appear to be inadequate for modeling our task.

Our generalized count approximation approach (Chapter 4) relaxes these limitations by using a hypergraph representation and establishing equivalent bi-directional transformations between function-free FOL and hypergraphs.

2.3 Robustness to systematic noise (sparse, low-quality observations)

2.3.1 Knowledge-based Learning

Using domain advice as inductive bias to accelarate learning has long been explored (Fung et al., 2002; Le et al., 2006; Towell and Shavlik, 1994; Kunapuli et al., 2010; Odom and Natarajan, 2018). Fu, (1995) presents a unified view of different variations of knowledge-based neural networks.

Such advice based learning has been proposed for support vector machines (Fung et al., 2002; Le et al., 2006) in propositional cases and probabilistic logic models (Odom and Natarajan, 2018) for relational cases. Towell & Shavlik, (1994) introduce the KBANN algorithm which compiles first order logic rules into a neural network and Kunapli et al., (2010) present the first work on applying advice, in the form of constraints, to the perceptron. The knowledge-based neural network framework has been applied successfully to various real world problems such as recognizing genes in DNA sequences (Noordewier et al., 1991), , robotic control (Handelman et al., 1990) and recently in personalised learning systems (Melesko and Kurilovas, 2018). Combining relational (symbolic) and deep learning methods has recently gained significant research attention. This is due to the fact that relational approaches are indispensable in faithful and explainable modeling of implicit domain structure, which is a major limitation in most deep architectures in spite of their success. While extensive literature exists that aim to combine the two (Sutskever et al., 2009; Rocktäschel et al., 2014; Lodhi, 2013; Battaglia et al., 2016), to the best of our knowledge, there has been little or no work on incorporating the advice in any such framework.

2.3.2 Modeling noise in deep models

Several recent approaches aim to make deep architectures robust to label noise by either learning from easy samples with importance weights or by additional noise-adaptation layers or, may be, by regularization over virtual adversarial randomization (Jiang et al., 2018; Goldberger and Ben-Reuven, 2017; Patrini et al., 2017; Miyato et al., 2018). While the above approaches enable effective learning of deep models in presence of noise, there are some fundamental differences with the problem setting of our knowledge-augmented deep model presented in Chapter 5.

(1) **[Type of noise]:** Our envisioned AI agent aims to handle *systematic* noise (Odom and Natarajan, 2018) which is frequent in real-world data due to cognitive bias in the data recording process or sample sparsity (refer Chapter, 1 Section 1.2).

(2) **[Type of error]:** Systematic noise leads to generalization errors (see Example 3).

(3) **[Structured data]:** Our work is in the context of structured data (entities/relations). Though crucial, structured data is inherently sparse (most relations are false in the real world).

(4) **[Noise prior]:** All the noise handling approaches for deep models mentioned earlier explicitly try to model the noise, which is impossible for systematic noise. Our Human-Allied AI setting instead allows expert knowledge to guide the learner towards better generalization via an inductive bias.

Augmented learning with human advice has been proven to be an effective strategy in machine learning, probabilistic learning or sequential decision making, in presence of systematic noise (sparsity + sample bias + errors in data recording). Although, pseudo-labels as weak supervision introduced by Lee, (2013) are used for constructing efficient semi-supervised methods in deep learning, weak supervision is not always successful as it assumes presence of large amounts of data and certainly not the best approach with noisy data (since the pseudo-lables are derived from the fully observed label set where noise could propagate). Advice is typically provided **before** the data set is encountered i.e., by a domain expert and hence is independent of the fully labeled data (which can be noisy). Data programming (Ratner et al., 2016) constrains the data using weak labels and is an orthogonal to our setting, since our approaches can be interpreted as constraining the model or hypotheses space.

2.3.3 Preference Elicitation in Sequential Decision Making (Planning)

As with predictive modeling (Sub-section 2.3.1), there has been a surge in interest towards using domain expertise to improve decision-making (Sohrabi and McIlraith, 2008; Sohrabi et al., 2009; Kunapuli et al., 2013; Judah et al., 2014; Odom and Natarajan, 2016a). While distinct methods differ in the form of knowledge that can be specified, *preference elicitation* has been explored inside automated planning (Myers, 1996; Boutilier et al., 1997; Huang et al., 1999; Brafman and

Chernyavsky, 2005; Sohrabi and McIlraith, 2008), reinforcement learning (RL) (Maclin and Shavlik, 1994, 1996; Natarajan and Tadepalli, 2005) and inverse reinforcement learning (IRL) (Kunapuli et al., 2013; Odom and Natarajan, 2016a).

The type of human knowledge that we aim to incorporate in our HAAI framework are represented as *preferences*. These correspond to IF-THEN statements where the IF defines the conditions (without negation) under which the preferences should apply and the THEN represents the preference. In RL or IRL, preferences could represent sets of preferred/non-preferred actions in a given set of states (Torrey et al., 2005). In the context of HTN planning, preferences could correspond to preferred/non-preferred decompositions of certain tasks (Sohrabi et al., 2009). Across these approaches, preferences have shown to be a good choice for specifying expert knowledge. However, many of these approaches require all of the preferences/advice upfront. This requires the domain expert (may not be machine learning experts) to provide the knowledge that would be useful for the learner. *Our approach solicits preferences during the decision-making process only as needed. This leads to effective interaction by reducing the number of uninformative queries*.

Recent work on active advice-seeking (Odom and Natarajan, 2016a) introduces a framework to allow the learning algorithm to request preferences over interesting areas of the feature or state space. To the best of our knowledge, our approach, oulined in Chapter 6, is the *first to actively solicit preferences in the planning setting*. Sarne et. al.'s work, on multi-agent planning & scheduling with mixture of human and autonomous agents (Sarne and Grosz, 2007), explicitly estimates the value of (future) additional information from humans about the feasibility of given options in order to generate queries. Our formulation does not assume existence of a "coordination" module making estimation of the value of information impossible. As a surrogate for such an estimate, PG-PLANNER (our framework presented in Chapter 6) instead identifies regions in task space where it has inadequate information. We show that our formulation facilitates human expert in providing more useful information, not limited to feasibility of given options.

Mixed-initiative planning (Ferguson et al., 1996; Ai-Chang et al., 2004; Talamadupula et al., 2013) interleaves planning by expert with automated planning which is similar in spirit to our

framework. However, they are conceptually different in the expert's role and intervention in the planning process. Mixed-initiative planning is a negotiation between an agent and human on mutable[3] goals/sub-goals, and partial plans. Such an intervention can be initiated by either party. On the other hand, our approach is responsible for only acquiring expert knowledge wherever it is expected to be most useful. Intuitively, in our setting, the responsibility to seek human intervention lies with the planner itself. since it allows the human to encode coarse knowledge and subsequently learn fine-grained knowledge via interaction.

[3] 'Mutable' refers to the fact that the sub-goals are not finalized before the planning process and may be updated.

PART I

OBSERVATION DENSITY: DATA-RICH ENVIRONMENTS

CHAPTER 3

SCALABLE LEARNING IN STRUCTURED NOISY ENVIRONMENTS

In this chapter we present our graph-based approach (Das et al., 2016) for approximating counts of satisfied instances of generalized patterns (First Order Logic rules) and show how probabilistic inference and learning can easily scale to arbitrarily large data sets by incorporating our approximation strategy.

3.1 Introduction

Statistical Relational AI(Getoor and Taskar, 2007) deals with the problems of learning and inference in the presence of rich, structured multi-relational data. A key operation required by most, if not all, StaRAI models is *counting*. For instance, Markov Logic networks (MLNs)(Domingos and Lowd, 2009) use counting as their fundamental operation in computing probabilities. Similarly, most directed probabilistic models employ a combination function such as mean or weighted mean(Natarajan et al., 2009) that require counts. Finally, lifted inference methods(Poole, 2003; Kersting et al., 2009; Milch et al., 2008) that aim to exploit symmetries during inference also require efficient counting.

However, since the counts are primarily used in exponents of different functions (be it the log-linear function of MLNs or the products in lifted inference methods), computing exact counts is not always necessary, especially when the counts are large. This is particularly true because most systems are relational where counting is one of the most expensive operations, more so, since counting all possible true relations is a major bottleneck (Poyrekar et al., 2014). For instance, intuitively, it should not matter (to an inference algorithm) whether a particular professor has co-authored approximately 500 publications or exactly 519 publications when answering a query about the success of this professor.

Our hypothesis is that efficiently performing approximate counting can allow for efficiency gains with a small loss of performance. To this effect, we exploit the progress in the graph

databases and graph theory (Metzler and Miettinen, 2015; Castellana et al., 2015; Sun et al., 2012) to perform fast, approximate counting over query structures. More specifically, we compile our logical model to a graph/network represented in *resource description framework* (RDF) format. This equivalent model allows for both approximate and exact counts to be performed in a fraction of time that is required by the original logical model.

We first show how the logical/relational model can be converted to an equivalent graph representation. Then, we present the exact computation algorithm that simply counts sub-graphs via queries. We then outline an approximation method (similar to message passing methods for probabilistic graphical models) that uses summary statistics (expected values) based on in-degrees and out-degrees to estimate the counts. Finally, we demonstrate the effectiveness and efficacy of the proposed counting approach on different types of problems in standard benchmark data sets.

To summarize, we make the following key contributions: (1) We propose a compilation strategy based on graph databases for relational probabilistic models. (2) We show how counting can be posed as queries in these models. (3) We derive a message passing method that can allow for fast approximate counts in such graphs. (4) Finally, we perform evaluation on several standard data sets in learning and probabilistic inference tasks.

3.2 Graph-Based Approximate Counting

We now present how formulas in first order logic (FOL)(Flach, 1994) can be represented as a "property graph" before presenting the counting algorithms. The key observation is that satisfiability of formulas in FOL (especially with fuction-free assumptions) is equivalent to 'searching' for the existence of a particular subgraph in the graph. Before we delve into further detail we present some necessary backgroung definitions:

Definition 1 (Property Graph (Model))**.** *Property Graph Model (Corby et al., 2000) is a model for representing graph structured data efficiently. Every edge* $\mathcal{E} = \langle v_1, v_2 \rangle$ *is denoted as a triple*

$\langle subject, predicate, object \rangle$, *such that,* $subject = v_1, object = v_2$ *and* $predicate = label(\mathcal{E})$. *Conceptually, predicate is a property of the subject, and the object is the value of that property. For example* $\langle Book, Name, \text{“}Hamlet\text{”} \rangle$ *is a triple where the predicate* $Name$ *is a property of the subject* $Book$ *and* $Hamlet$ *is the value of the property* $Name$. *This allows us to encode multiple types of entities and relations between entities on a single graph.*

Definition 2 (Resource Description Framework (RDF))**.** *Resource Description Framework (RDF) (Corby et al., 2000) is a framework for representing a property graph where subject, predicate or object is a resource, with a namespace binding. The namespace has to be a valid URL.*

Definition 3 (SPARQL)**.** *SPARQL (PrudHommeaux et al., 2008) is a query language for querying RDFs. It is different from SQL, in that the subgraph we query is encoded in the WHERE part of the query as connected triples. The variables are denoted with a '?' in front and the constants must be bound by the declared namespace as shown below.*

PREFIX namespace:$\langle url \rangle$ *SELECT ?a ?b ?c FROM* G_f
WHERE {*?a namespace:Friends ?b. ?b namespace:Hates ?c*}

3.2.1 Structure transformation

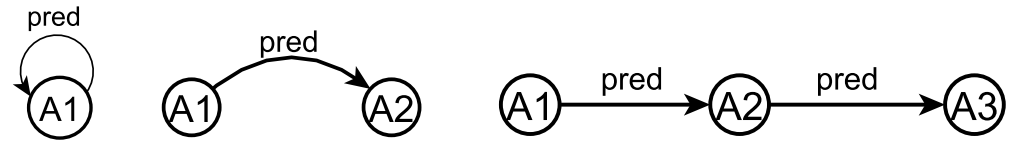
Given the predicate type definitions in logical form, the goal is to construct an equivalent property graph $\mathcal{G}$. The arguments of the predicates become the nodes of $\mathcal{G}$. Each predicate becomes a labeled, directed edge in $\mathcal{G}$ with the label being the name. This edge connects the nodes which are the argument of the corresponding predicate. The direction of the edge is determined by the order of the arguments of the predicate. This is motivated by the fact that predicates express relation between two or more entities or at times a property of an entity, viewed as a reflexive relation of an entity with itself.

For example consider a predicate $pred(A_1, A_2)$. The arguments A_1 and A_2 are added to nodes set ($\mathcal{N}$) in $\mathcal{G}$. $\mathcal{E} = A_1 \rightarrow A_2$ is a directed edge added to the set of edges E, where $label(\mathcal{E}) = pred$. We now show how we handle the predicates.

- *Unary Predicates:* For unary predicates ($pred(A_i)$), the directed edge will be a 'self-loop', i.e., $\mathcal{E} = A_i \rightarrow A_i \in E$. This is illustrated in Figure 3.1(a).

- *Binary Predicates:* This is the straightforward case. For $pred(A_1, A_2)$, the two nodes A_1 and A_2 have a directed edge $\mathcal{E} = A_1 \rightarrow A_2 \in E$. See Figure 3.1(b).

- *N-ary Predicates:* The more general case can possibly be handled by converting the N-ary to $N - 1$ binary predicates as is done with most formalisms. Thus, for predicate $pred(A_1, A_2, A_3)$, we can construct two edges $\mathcal{E}_1 = A_1 \rightarrow A_2$ and $\mathcal{E}_2 = A_2 \rightarrow A_3$ (Figure 3.1(c)), both the edges having the same label. While this can introduce some spurious relations (Kersting and De Raedt, 2007), with large amounts of data, this might be a reasonable approximation. However, in the current context, we focus on only unary and binary cases. Handling the N-ary predicates in a principled manner is presented in the next chapter.

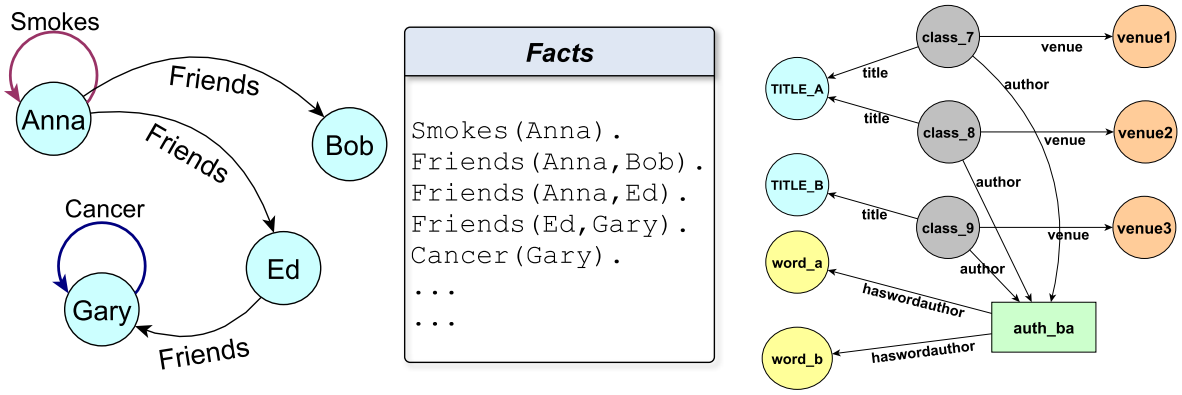


(a) Unary (b) Binary (c) N-ary

Figure 3.1: Handling Arity

To summarize, given a set of facts (evidence) $F_{ev} = \{pred_i(A_{i1}, A_{i2}, \ldots, A_{ij})\}_{i=1}^{K}$ where K is the number of facts and $j \geq 1$ (value of j is the Arity of predicate $pred_i$), we construct a corresponding evidence graph G_{ev} of size $O(K * |A|)$. We use the standard *"Smokes-Friends-Cancer"* problem (Domingos and Lowd, 2009) and demonstrate the graph construction in Figure 3.2(a). As can be observed, $Smokes(Anna)$ and $Cancer(Gary)$ are both unary predicates and hence are self-loops while others are directed edges. As another example, for the facts (about authors and venues) presented below, the equivalent graph is shown in Figure 3.2(b).

$$author(\text{“}class\_7'', \text{“}auth\_ba'').\ author(\text{“}class\_8'', \text{“}auth\_ba'').\ author(\text{“}class\_9'', \text{“}auth\_ba'').$$
$$title(\text{“}class\_7'', \text{“}TITLE\_A'').\ title(\text{“}class\_8'', \text{“}TITLE\_A'').\ title(\text{“}class\_9'', \text{“}TITLE\_B'').$$
$$venue(\text{“}class\_7'', \text{“}venue1'').\ venue(\text{“}class\_8'', \text{“}venue2'').\ venue(\text{“}class\_9'', \text{“}venue3'').$$
$$haswordauthor(\text{“}auth\_ba'', \text{“}word\_a'').\ haswordauthor(\text{“}auth\_ba'', \text{“}word\_b'').\quad (3.1)$$



(a) Smokes Friends Graph (b) Author-Venue Property Graph

Figure 3.2: Property Graphs for Smokes-Friends *(left)* and Author-Venue *(right)* domains

Algorithm 1 presents the creation of the evidence graph. Evidence graph construction is straightforward parsing of facts (ground atoms given as evidence) and adding corresponding nodes and edges to the graph G_{ev} as described earlier. It runs in linear time $O(n)$, n being the size of the evidence. However, arity of the predicates have to be handled carefully.

3.2.2 Obtaining approximate satisfiability counts

We now explain the counting process given summary statistics of a graph. The key is that this is equivalent to counting subgraphs in a heterogeneous network while satisfying certain constraints. For example, consider the Smokes-Friends-Cancer example in Figure 3.2(a). For simplicity, let us assume the following clause: $Smokes(a1) \wedge Friends(a1, a2)$.

Algorithm 1 CreateGraph

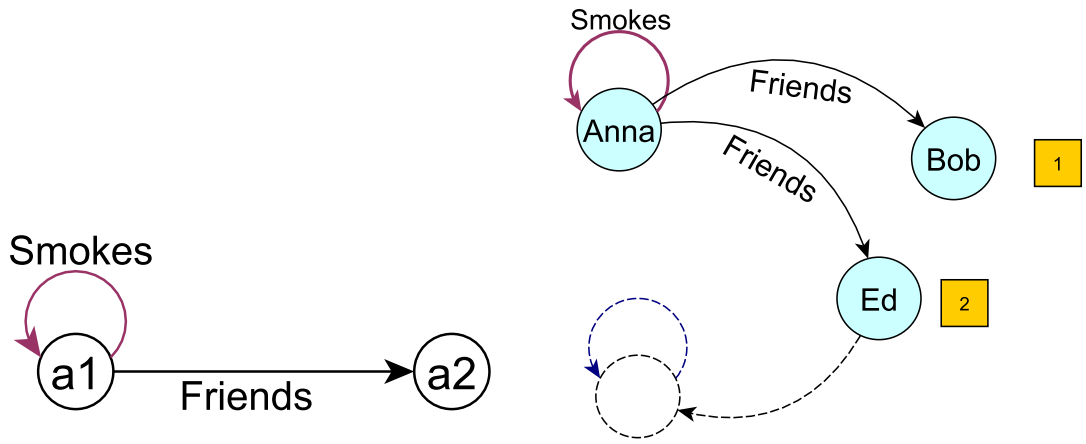
procedure CREATEGRAPH($\mathcal{F}$)
    Input: Evidence File $\mathcal{F}$
    Output: Evidence Graph $\mathcal{G}_{ev}$
    Initialization: Empty Evidence Graph $\mathcal{G}_{ev} = \{\}$
    for each ground atom $\mathcal{P} = p(A_1, A_2, ..)$ in $\mathcal{F}$ **do**
        Parse $\mathcal{P}$
        Edge $\mathcal{E} \leftarrow p$
        if $\mathcal{P}$ is unary **then** ▷ transforming unary predicates into loop edge
            Subject Node $\mathcal{N}_s \leftarrow A_1$
            Object Node $\mathcal{N}_o \leftarrow \mathcal{N}_s$ [Self loop]
        else if $\mathcal{P}$ is binary **then** ▷ transforming binary predicates into typical edge
            Subject Node $\mathcal{N}_s \leftarrow A_1$
            Object Node $\mathcal{N}_o \leftarrow A_2$
        end if
        New Triple $\mathfrak{T} \leftarrow \langle \mathcal{N}_s, \mathcal{E}, \mathcal{N}_o \rangle$ ▷ creating RDF triple for storage
        Add $\mathfrak{T}$ to $\mathcal{G}_{ev}$
    end for
return ($\mathcal{G}_{ev}$)
end procedure

To calculate the number of satisfiable groundings of this clause, we count the subgraphs (motifs) that have the structure shown in Figure 3.3(a) that are present in the Smokes-Friends-Cancer network given in Figure 3.2(a). This particular structure becomes the constraint on the counting task, i.e., the goal is to count subgraphs that satisfy this (structure) constraint. Figure 3.3(b) is an example of this structure given two groundings of the above clause (Bob and Ed being friends of Anna). i.e.,

$\mathbf{1} : Smokes(Anna) \wedge Friends(Anna, Bob); \quad \mathbf{2} : Smokes(Anna) \wedge Friends(Anna, Ed)$

Exact Counting

Given that we have mapped the counting of satisfied groundings to subgraph counting, exact counting can be performed in a relatively straightforward manner. First, we represent the property graph as an RDF (as shown earlier). Now, exact counting requires simply retrieving all the subgraphs matching the motif or the pattern induced by a clause as shown in Figure 3.3 and finally, enumerating and counting them. This can be achieved by a straightforward SPARQL query.



(a) First Order Clause (b) SubGraphs satifying the clause

Figure 3.3: FO-Clause and equivalent subgraph counting

For the clause $Smokes(a1) \wedge Friends(a1, a2)$, the query to return all the subgraphs is:

```
SELECT ?a1 ?a2 FROM Evidence_Graph
                WHERE {?a1 Smokes ?a1. ?a1 Friends ?a2}
```

This query will return all the subgraphs present in the evidence graph G_{ev} which can then be counted. All the constraints (here parameterised first order predicates) are encoded in the "WHERE" part of the query. Once all possible subgraphs in the evidence graph are returned into a result set $R_s = \{g_s^{(1)}, g_s^{(2)}, \ldots\}$, the size of result set $|R_s|$ is the count value.

While this is straightforward, we have just essentially converted one NP-Complete problem to another, since sub-graph matching is already a hard problem. Even though databases might allow for efficient counting, this may not be tractable for all problems. So we next present an approximate technique.

Approximate counting

We now present our algorithm called FACT (**F**ast **A**pproximate **C**oun**T**ing for relational probabilistic models). Our key intuition remains the same – a clause is equivalent to a pattern and our goal is

Algorithm 2 Summarize

procedure SUMMARIZE($\mathcal{G}_{ev}$)
    Input: Evidence Graph $\mathcal{G}_{ev}$
    Output: Summary statistics in a set of Hash data structures $\{H_{in}, H_{out}, H_{in}^{(avg)}, H_{out}^{(avg)}\}$
    Initialization: $\{H_{in}, H_{out}\}$ as empty structures.
    $Query \leftarrow$ "SELECT {count(?s) as ?cnt} ?p ?o from $\mathcal{G}_{ev}$ WHERE {?s ?p ?o} GROUP BY ?p ?o"
    ResultSet $R_s \leftarrow execute(Query, G_{ev})$
    for each $\langle cnt, p, o\rangle \in R_s$ **do**
        $put(H_{in}, \langle o, \langle p, cnt\rangle\rangle)$
    end for
    $Query \leftarrow$ "SELECT ?s ?p {count(?o) as ?cnt} from $\mathcal{G}_{ev}$ WHERE {?s ?p ?o} GROUP BY ?s ?p"
    ResultSet $R_s \leftarrow execute(Query, G_{ev})$
    for each $\langle s, p, cnt\rangle \in R_s$ **do**
      $put(H_{out}, \langle s, \langle p, cnt\rangle\rangle)$
    end for
    $H_{in}^{(avg)}, H_{out}^{(avg)} \leftarrow Aggregate_{predicates}^{(average)}(H_{in}, H_{out})$
return $H_{in}, H_{out}, H_{in}^{(avg)}, H_{out}^{(avg)}$
end procedure

to search for that pattern in the evidence graph G_{ev} as shown in Figure 3.3. The important difference is that instead of getting the exact counts of the sub-graphs, we propagate summary statistics obtained via the *Summarize()* procedure in Algorithm 2 to estimate the counts of the query results. Summarization (Algorithm 2) involves getting the in-degree and out-degree summaries of each node in the graph G_{ev} via aggregation queries (SPARQL) and storing them using in-memory data structures. Table 3.1 shows an example summary table based on the graph shown in Figure 3.2(a).

Table 3.1: Example summary

Node	Edge Label (Predicate)	Out-Degree
Anna	Smokes	1
	Friends	2
Ed	Friends	1
Gary	Cancer	1

Inspired by the success of message passing algorithms in probabilistic graphical models, we develop an algorithm that uses augmented count values from summary statistics as messages. Before presenting the algorithm, we define a few terms.

- $In_{pr}(c)$ is the in-degree of a node with constant c present in G_{ev} w.r.t edges with predicate pr. In Figure 3.2(b), $In_{author}(auth\_ba) = 3$.

- $Out_{pr}(c)$ is the out-degree of a node with constant c present in G_{ev} w.r.t edges with predicate pr. In Figure 3.2(b), $Out_{hasw ordauthor}(auth\_ba) = 2$.

- $In_{pr}^{(avg)}$ is the average in-degree of all nodes that have incoming edge $\mathcal{E}$ with $label(\mathcal{E}) = pr$. Hence $In_{pr}^{(avg)} = \frac{\sum_i^N In_{pr}(v_i)}{N}$.

- $Out_{pr}^{(avg)}$ is the average out-degree of all nodes that have incoming edge $\mathcal{E}$ with $label(\mathcal{E}) = pr$. Hence $Out_{pr}^{(avg)} = \frac{\sum_i^N Out_{pr}(v_i)}{N}$.

- $\Theta(v)$ or type count is the number of constants possible (given by the evidence) for variable v. So for parameterized predicate $pr(v)$, if the structure of the predicate is $pr(type_A)$, i.e., the argument of the predicate is of type $type_A$ then $\Theta(v) = |A|$, where $type_A \leftarrow A = \{a_1, a_2, \ldots\}$.

- $\mu_{v_i \rightarrow v_j}^{pr}$ is the message transmitted from node (variable) v_i to node v_j over edge pr.

- $\mathcal{G}_q$ is the query graph, or the graph formed by the formula/clause.

Next we will first describe the process informally, and present the algorithm. For example, consider the clause below whose equivalent graph for the body of the clause[1] is shown in Figure 3.4.

$$pr1(a_1, a_2) \wedge pr2(a_3, a_2) \wedge pr3(a_2, a_4) \Rightarrow h(a_2) \tag{3.2}$$

We start with the assumption that at least one of the variables is grounded, that is, the counts are obtained against one value of the variable (in our case let us assume that the variable $a2$ is

[1]This is an example of a horn clause of the form a implies b. a is the body (antecedent) and b is the head (consequent).

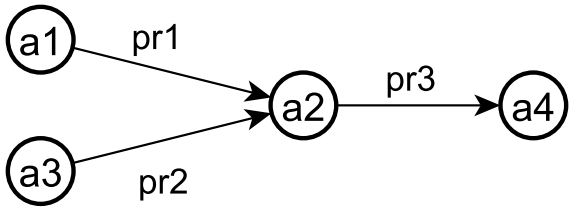
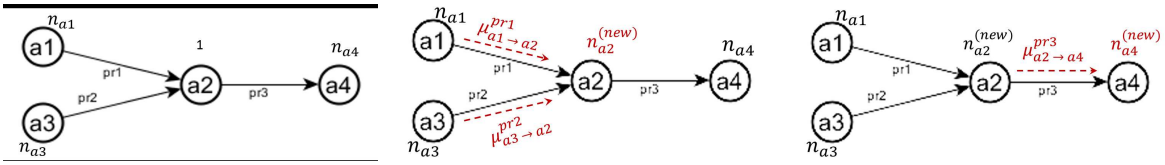


Figure 3.4: Graph for clause in Equation 3.2



(a) Initialization (b) Next step (c) Final step

Figure 3.5: Approximate Counting

grounded). This is typically the case in several problems. For instance, during probabilistic inference, we usually condition the query on the value of a variable. Or when learning the parameters, we learn the distribution over the children values given the values of the parent. This value is what we refer to as being grounded. However, even if no variable is grounded, we can always sum over the counts for all values of a variable in the clause.

With this assumption, we first initialize the counts of each variable, in the graph formed using the body of the clause (Figure 3.4). As the variable $a2$ is grounded, its count is initialized to 1 and all other variables are initialized to their type counts, $\Theta(v)$. This is illustrated in figure 3.5(a). At the initial state, $count(a1) = n_{a1} = \Theta(a1)$, $count(a2) = 1$ $[\because a2 \; is \; a \; constant]$, $count(a3) = n_{a3} = \Theta(a3)$ and $count(a4) = n_{a4} = \Theta(a4)$.

Given that the graph is initialized, we now demonstrate how the message passing occurs here and the counts are updated. One important factor here is that, the graph is directional hence the order of the variables (for eliminating variables during counting) is straightforward. The variables (nodes) that have no incoming edges in the query graph $\mathcal{G}_q$ form the starting nodes for propagation

and the order of the rest is implied. Thus, in our example we start with $a1$ and $a3$ and messages are passed from these nodes to node $a2$ [$\mu^{pr1}_{a1\rightarrow a2}$ from $a1$ to $a2$ and $\mu^{pr2}_{a3\rightarrow a2}$ from $a3$ to $a2$] as displayed in Figure 3.5(b). The messages are simply augmented counts:

$$\mu^{pr1}_{a1\rightarrow a2} = \frac{Out^{(avg)}_{pr1}}{\Theta(a2)}.\frac{In_{pr1}(C)}{\Theta(a1)}.n_{a1} \tag{3.3}$$

The expression $\frac{Out^{(avg)}_{pr1}}{\Theta(a2)}$ gives us the ratio of the average counts of out-going edges to the maximum number of possible outgoing edges, with predicate $pr1$ in this case. $\frac{In_{pr1}(C)}{\Theta(a1)}$ is a similar expression for the case of the incoming edge. Their product gives us an approximation of the *expected counts* of the predicate in G_{ev}, which we then use to augment the count. Similarly, the message $\mu^{pr2}_{a3\rightarrow a2}$ is obtained as,

$$\mu^{pr2}_{a3\rightarrow a2} = \frac{Out^{(avg)}_{pr2}}{\Theta(a2)}.\frac{In_{pr2}(C)}{\Theta(a3)}.n_{a3} \tag{3.4}$$

Now, the count value of $a2$ is updated. Note how the mean/average of in-degree is not considered here, since the variable $a2$ is grounded (constant C).

$$n^{(new)}_{a2} = \prod_{(v,pr)\in\{(a1,pr1),(a3,pr2)\}} \mu^{pr}_{v\rightarrow a2}.n^{(old)}_{a2} \tag{3.5}$$

$where\ n^{(old)}_{a2} = 1$. Finally another message $\mu^{pr3}_{a2\rightarrow a4}$ is passed from $a2$ to $a4$ as shown in figure 3.5(c). The message is as shown below:

$$\mu^{pr3}_{a2\rightarrow a4} = \frac{Out_{pr3}(C)}{\Theta(a4)}.\frac{In^{(avg)}_{pr3}}{\Theta(a2)}.n^{(new)}_{a2} \tag{3.6}$$

Updating the count of $a4$ works in a similar fashion as shown in Equation 3.5. Due to reasons mentioned earlier, mean/average out-degree is not considered (in Eqn 3.6), instead the out-degree of the exact constant node is used.

Given $n^{(new)}_{a4}$, it is the final count that is required, since it is the only variable left after eliminating the rest, which is an approximate estimate of the number of subgraphs present in G_{ev}. We now formally present the algorithm FACT (Algorithm 3[2]).

[2]Note: In Algorithm FACT *parse()* is just a name given (for ease of representation) to the operation of parsing an FOL clause into a query graph as shown in Equation 3.2 and Figure 3.4

Pre-processing: To preprocess and construct the graph, we call methods $CreateGraph(\mathcal{F})$ (*Algorithm 1*) to get the evidence graph G_{ev} and call $Summarize(G_{ev})$ (*Algorithm 2*) to get the summary data structures $H = \{H_{in}, H_{out}, H_{in}^{(avg)}, H_{out}^{(avg)}\}$ at the beginning of any inference or learning system.

Algorithm 3 FACT

procedure FACT($\mathfrak{C}, H, C, \Upsilon$)
    Input: Clause $\mathfrak{C}$,H, constant C for variable Υ
    Output: Approximate count cnt
    Initialization: Build $G_q(N_q, E_q) \leftarrow parse(\mathfrak{C})$
    $n(u_i) \leftarrow \Theta(u_i) : u_i \in N_q - \Upsilon$ and $n(\Upsilon) \leftarrow 1$
    Start eliminating nodes with no incoming edge in G_q
    $V \leftarrow (\{v : v \in N_q\} - \{n : n \in N_q; \nexists(u \curvearrowright n \in E_q)\})$
    for each variable $v \in V$ **do**
        for $x \in N_q$, s.t. $\exists pr(x, v) : x \curvearrowright_{pr} n \in E_q$ **do**
            if $v = \Upsilon$ (substitute constant) **then**
                $\mu_{x \to v}^{pr} \leftarrow \frac{Out_{pr}^{(avg)}}{\Theta(v)} . \frac{In_{pr}(C)}{\Theta(x)} . n_x$
            else
                if $x = \Upsilon$ (substitute constant) **then**
                    $\mu_{x \to v}^{pr} \leftarrow \frac{Out_{pr}(C)}{\Theta(v)} . \frac{In_{pr}^{(avg)}}{\Theta(x)} . n_x$
                else
                    $\mu_{x \to v}^{pr} \leftarrow \frac{Out_{pr}^{(avg)}}{\Theta(v)} . \frac{In_{pr}^{(avg)}}{\Theta(x)} . n_x$
                end if
            end if
        end for
        $n_v^{(new)} \leftarrow \prod_{\{(x_i, pr(x_i, v))\}_i} \mu_{x \to v}^{pr} . n_v$
        if $sizeOf(V) = 1$ **then**
            $cnt \leftarrow n_v$
            break
        end if
        $V \leftarrow V - \{v\}$
    end for
return cnt
end procedure

Discussion

There are a few important things to mention before presenting the experiments. First, we apply the *Closed World* assumption, which allows us to model the negation of a predicate as absence of an edge. Also, for a self loop, degree summary is considered to be either 1 or 0.

Second, for simplicity and ease of representation, we consider horn clauses i.e., clauses with only 'conjunction' ($\wedge$) operators in the body (Eqn 3.7).

$$\wedge_i pred_i^{(body)}(a_1, \ldots, a_x) \implies pred^{(head)} \tag{3.7}$$

Third, and most importantly, to prove that the values returned by our message-passing/variable-elimination based system can be considered as counts, it is important to note a few characteristics of the algorithm FACT: (1) If we multiply the initial count values of the variables in a query graph, such as in Figure 3.5(a), we will get the size of the full cross-product. (2) Instead, the messages "augment" these counts with the 'belief' about the presence of particular predicate/edge, based on G_{ev}, hopefully bringing the expected value closer to the exact count. (3) The counts returned by our method are, often, slight over-estimations. However, as we show empirically, these are reasonable approximations that can be computed in a fraction of the time required for precise counting.

3.3 Implementation & Experimental Evaluation

To seamlessly handle multi-relational graphs we employ a powerful graph representation language, RDF (Corby et al., 2000; Pan, 2009), from the Graph Database community. We also employ the SPARQL query language for querying on Graph structured data represented via RDF. Our Java implementation uses "Apache Jena" a Java Library that provides an API which allows for fine grain manipulation of Graph Structured data represented in RDF form and also supports SPARQL 1.1 (the latest version of query semantics on RDF). Note that off-the-shelf graph database systems have their internal optimizers making it hard to benchmark the effectiveness of the proposed approach by simply employing them. Hence, we use Apache Jena. Finally, we assume the inputs to be in predicate logic format, since this is the common representation used across many SRL models.

3.3.1 Experimental design

Our experiments, aim to answer the following questions:

(**Q1:**) How effective is the proposed approach compared to the standard counting method?

(**Q2:**) How does approximation affect performance of the learned/inferred models?

(**Q3:**) Is the proposed approach useful across a variety of learning and inference tasks inside Statistical Relational Learning (SRL)?

Motivated by **Q3**, we evaluate our algorithm in three different types of SRL tasks - parameter learning with combining rules, full model (structure+ parameters) learning from labeled data and finally, performing lifted probabilistic inference that counts symmetric groups of variables when answering probabilistic queries. For each of this setting, we used the corresponding state-of-the-art algorithm and replaced the counting computations inside them with our approach. We discuss each of them in detail.

Datasets: We primarily use several standard SRL data sets for evaluation, namely (1) ***Yeast*** data set, which is about interaction among proteins, protein complexes and enzymes in yeasts. The goal is to predict the class of protein. (2) ***IMDB*** data set, mainly about actors, directors and movies, where the goal is to predict $workedUnder(person1, person2)$, (3) ***Cora***, where the primary goal is to predict if two citations are the same (particulary if the venues are same, i.e., predict $samevenue(venue1, venue2)$ in case of our experiments), (4) ***WebKB***, where the goal is to predict $departmentOf(webPage, webPage)$ predicate - i.e., department of a web page, and (5) ***Smokes-Friends-Cancer*** data set (Domingos and Lowd, 2009), where the goal is to predict who has cancer based on the friends network of individuals and their observed smoking habits. Note that all the results presented in the system are the results of 5 runs, but the sizes of the datasets vary according to the problem. We present the appropriate sizes in the results.

Table 3.2: Results of approximate counting on Combining Rules

Domains	# Facts	# Clauses	Metrics	CombRulesApprox	CombRulesOriginal
yeast	819	1600	Counting time (secs)	2.80 ± 0.01	7.89 ± 0.12
			MSE	0.26	0.24
			CLL	-0.72	-0.67
IMDb	264	16	Counting time (secs)	0.19 $\pm 1.00E-04$	0.36 $\pm 1.00E-04$
			MSE	0.17	0.09
			CLL	-0.58	-0.23
Cora	1498	75	Counting Time (secs)	0.44 $\pm 5.62E-05$	1.28 $\pm 6.87E-05$
			MSE	0.22	0.22
			CLL	-0.64	-0.63
WebKB	12284	11K	Counting Time (secs)	4.84 $\pm 6.67E-11$	8.41 $\pm 4.55E-10$
			MSE	0.29	0.29
			CLL	-0.63	-0.59

3.3.2 Task 1: Learning Parameters of Relational Models

To demonstrate the usefulness of our proposed approach on learning probabilistic relational models, we consider the problem of parameter learning with mean and weighted-mean combining rule (Natarajan et al., 2009). Specifically, we consider the use of EM for learning as developed by Natarajan et al. (Natarajan et al., 2009). The key step of this algorithm is to count the number of satisfied groundings of a clause and the number of satisfied groundings across clauses that share the same target predicate. We replace their simple counting method (*CombRulesOriginal* in the results) with our approximate counting method (*CombRulesApprox* in the results). We used 4 bench-mark data sets (1) ***Yeast***, (2) ***IMDB***, (3) ***Cora*** and (4) ***WebKB***. Their sizes are presented in Table 3.2.

For both approaches, we measure the following - (1) *Counting time*: Time taken for counting groundings that satisfy a clause for every example. (2) *MSE*: Mean Squared Error. (3) *CLL*: Conditional Log Likelihood. Table 3.2 presents the results of our experiments. It can be observed that there is at least a $2-3$ times reduction in counting time over all the data sets when using our approximate counting approach. The MSE $mse(CombRulesApprox)$ is comparable to the original MSE $mse(CombRulesOriginal)$, i.e., MSE does not suffer much because of approximation. However as can be seen for the *IMDB* data set, the performance seems to be much worse than

the original approach. Our hypothesis is that the data set is too small and since our approximate counting mechanism is based on sample averages, they do not serve as good approximations on small data sets. However, with larger data sets, the efficiency gains are higher with smaller losses in effectiveness. Thus **Q1** and **Q2** can be answered by observing that for reasonably larger data sets, the proposed method is effective and efficient.

3.3.3 Task 2: Learning Structure of SRL Models

Table 3.3: Results of Approx. Counting on MLN-Boost

Domains	#Facts	Metrics	MLN-BoostApprox	MLN-BoostOriginal
Yeast	1641	Learning Time(millisec)	15040 $\pm$2.89	34108 $\pm$3.45
		Inference Time(millisec)	749 $\pm$42.88	1346 $\pm$67.02
		AUC ROC	0.5	0.51
		AUC PR	0.38	0.45
WebKB	26223	Learning Time(millisec)	10609 $\pm$102.18	73123 $\pm$364.7
		Inference Time(millisec)	1542 $\pm$96.55	3915 $\pm$80.01
		AUC ROC	1.00	1.00
		AUC PR	1.00	1.00
Smokes-Friends-Cancer	150,401	Learning Time(millisec)	34K $\pm$199.56	114K $\pm$402.09
		Inference Time(millisec)	806 $\pm$201.55	2183 $\pm$189.61
		AUC ROC	0.74	0.75
		AUC PR	0.82	0.82

Given the performance in parameter learning, we next turn our attention to learning structure of SRL models. Specifically, we consider the state-of-the-art learning method for Markov Logic Networks that employs Functional Gradient Boosting (Khot et al., 2011) (called MLNBoost). Markov Logic networks (Richardson and Domingos, 2006), in brief, are weighted first-order logic clauses that soften logic by allowing the clauses to be unsatisfied in some cases. Roughly, the probability of a given world (set of facts) being true is proportional to the weigted count of the satisfied groundings of all the clauses. Specifically, the probability distribution over possible worlds x specified by a ground Markov network is given by $P(X = x) = \frac{1}{Z} \exp\left(\sum_i W_i n_i(x)\right)$ *(where $n_i(x)$ is the number of true groundings of the first-order formula F_i in x, $x_{\{i\}}$ is the state (truth values) of the atoms appearing in F_i).* We approximate $n_i(x)$. Count approximation is significant here since,

complete grounding (instantiation) is a major bottleneck in inference tasks for MLNs, especially when evidence is large (Poyrekar et al., 2014). Nearly all learning methods implicitly use inference in their inner loop, making count approximation a compelling improvement strategy.

As with the previous experiment, we replace the counting of non-trivial groundings of a clause (Khot et al., 2011) of the original system (called *MLN-BoostOriginal* in results), with our efficient approximation method (referred as *MLN-BoostApprox* in results). We used 3 data sets (1) *Yeast*, (2) A larger version of *WebKB* and (3) *Smokes-Friends-Cancer* (Table 3.3).

Since, inference is used inside MLN structure learning, we employed approximate counting method for both inference and learning. We measured the following metrics to compare *MLN-BoostOriginal* and *MLN-BoostApprox* : (1) Learning Time, (2) Inference Time, (3) AUC-ROC and (4) AUC-PR. Table 3.3 presents the results of the experiment. It was observed that our method brings reasonable improvement in Learning as well as Inference time without any significant deterioration in AUC-ROC/AUC-PR values. This allows us to answer both **Q1** and **Q2** affirmatively, and our method being more efficient than the state-of-the-art on these challenging tasks makes for a compelling case.

3.3.4 Task 3: Lifted Inference

As a final task, given the recent surge in interest in the so-called **lifted probabilistic inference** methods, we employed our approximation strategy for counting in one such method. Specifically, we considered C-FOVE (Milch et.al.)(Milch et al., 2008) that counts over satisfied formulas when grouping evidence into symmetrical sets. We chose this method primarily because : (1) A working implementaion of C-FOVE (on Java platform) is easily available online and (2) It is illustrative of how lifted inference methods need counting for them to be efficient. Specifically, this method focuses on explicit counting of satisfied formulae making it possible for our approximate method to directly replace their counting procedure.

We were able to run both the Original C-Fove system and our customized system (with approximation) using *Smokes-Friends-Cancer* dataset presented earlier. However, in this experiment, we had to consider a restricted size of 800 facts and 30 constants. Datasets larger than this, caused Java Heap Memory issues in the original system. The results are presented in Table 3.4, showing (1) Reasonable gain in *Counting Time* $\mathbf{t}$ due to approximation and (2) Negligible *Average Absolute Difference (AAD)*, between the exact *final probability values* (computed by C-Fove) and the ones via approximation, for a given set of 10 query predicates. The results are similar to previous ones, however the efficiency decrease is lower. A potential reason for this is the restricted representation used by the C-FOVE system. The system is restricted to single length counting formulas where counting is already reasonably efficient. Despite this, our system demonstrates a 50% decrease in learning time compared to original system. This clearly answers **Q1** and **Q2** affirmatively.

Table 3.4: Results of Approx Counting on C-Fove

	#Facts	**Metrics**	**Approx**	**Original**
Smokes	800	$\mathbf{t}$ (sec)	4.11 $\pm$0.02	6.72 $\pm$0.00467
		AAD	0.016	0.0

Final note: Given our experiments, **Q3** can be answered affirmatively, that our proposed approach is both effective (predictive performance) and efficient (running time) across learning and inference tasks against state-of-the-art systems in benchmark data sets.

3.4 Conclusion

We presented the first compilation to graph data bases method for approximate counting of satisfied instances - a crucial operation in several relational probabilistic models. It converts the predicate logic format to an equivalent graph database format that can be queried effciently. Our novel approximate counting method is inspired by probabilistic message passing, and our extensive experimental results demonstrate that it is effective in learning both parameters and structure and performing probabilistic inference from moderate to large data sets, while reducing the running time significantly.

CHAPTER 4

GENERALIZED COUNT APPROXIMATION

The count approximation strategy detailed in the previous chapter, although successful in many tasks, has several fundamental limitations. Such limitations exist both due to our representational choice and solution formulation. This makes it challenging for our approach to be adapted to several types of tasks and domains. In this chapter we address those challenges and present a generalized approach, MACH (Das et al., 2019a), for approximating counts of satisfied instances (groundings).

4.1 Introduction

Significant advancements in research on Statistical Relational Learning (SRL) and AI (Getoor and Taskar, 2007; Raedt et al., 2016) and in lifted inference (Poole, 2003; Kersting et al., 2009) have allowed for exploiting the symmetries of the data and model during learning and inference. The advantage of these algorithms is that they can succinctly represent and reason with dependencies among the attributes and relations of related objects. One of the key operations inside most, if not all, algorithms is *counting* the satisfied groundings (instances) of a partially instantiated relational rule (a first-order clause). For instance, when learning the parameters or structure of a Markov Logic Network (MLN) (Domingos and Lowd, 2009; Khot et al., 2011), or when performing lifted inference (Poole, 2003) one has to compute the expected/true counts in the model/data and inside a given cluster of objects.

Counting is a hard combinatorial search problem ($\#P$-complete). Consequently, algorithms for fast, approximate counting have been developed (Das et al., 2016; Sarkhel et al., 2016). *The key observation is that computing exact counts is not essential, particularly when the number of groundings for an object/relation is high*. For instance, knowing whether a Professor published 300 papers or 319 papers does not significantly change the belief over the popularity of the Professor.

While reasonably successful, they make a few assumptions – including MLN-specific formulation and lack of support for partial groundings (Sarkhel et al., 2016) or restricted arity of relations (Das et al., 2016).

We relax these assumptions and present a *general approximate counting* technique that transforms the problem of counting partially instantiated clauses to motif-matching in equivalent hypergraph. Provably, counting in the original data corresponds to **computing *the expected counts* of the motif occurrences** in the transformed hypergraph: When this expectation is computed exactly, one can retrieve the true counts. In large data sets, this motivates the approximation of the expectation using summary statistics on the hypergraph. Our *experimental results across several domains on both learning and inference tasks* demonstrate clearly that this approximation indeed relaxes the assumptions of the previous methods and results in more efficient counting while exhibiting on-par performance to exact counting.

This chapter makes the following contributions - (1) We present a method for converting structured data and relational/logic clauses to hypergraphs and motifs, respectively, and pose the problem of counting the number of groundings as counting over motifs-matches in the hypergraph. (2) We present and justify an approximation technique and outline the algorithm for counting the number of motifs based on the order of the variables that appear in the clauses. (3) Finally, we demonstrate the efficiency of the proposed approach without sacrificing the effectiveness on standard domains against state-of-the-art baselines on these tasks.

The rest of the chapter is organized as follows - we introduce the necessary notations, explain the conversion process and outline our proposed algorithm. We then present our empirical evaluations before concluding the chapter by outlining the areas of future research.

4.2 Motif-based Approximate Counting via Hypergraphs

Our goal is to compute the counts of a potentially partially instantiated clause given a database of ground assertions. We proceed by transforming a SRL model into a directed graph notation. Trivial

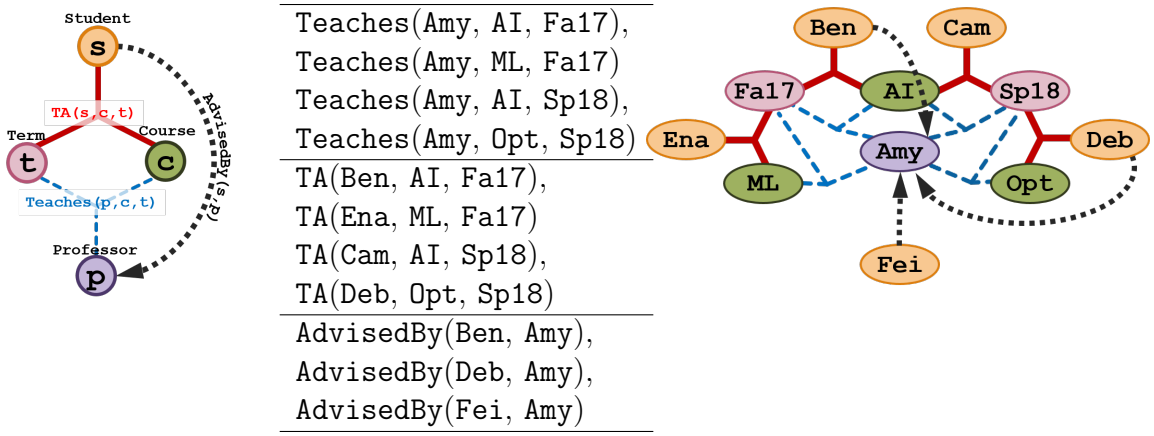


Figure 4.1: (**left**) Motif $\mathcal{M}_1$ for $\mathtt{C}_1$; (**center**) Facts used to ground $\mathcal{M}_1$; (**right**) Ground graph, $\mathcal{G}_1$. Ternary predicates `Teaches` and `TA` are represented as hyperedges in both $\mathcal{M}_1$ and $\mathcal{G}_1$. The edges `AdvisedBy(Deb, Amy)` and `AdvisedBy(Fei, Amy)` also appear in the grounding of $\mathtt{C}_2$ (Fig. 4.2).

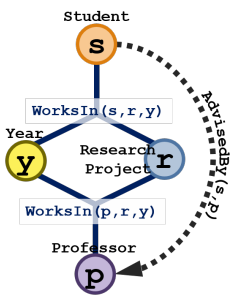
conversion from a logic statement (essentially a *conjunctive rule*) to a simple directed graph has an important limitation of assumimng binary relations. Such graphs, however, cannot represent n-ary relations, which are very common. We employ hypergraphs (Berge and Minieka, 1973), generalization of graphs in which a hyperedge joins an arbitrary number of nodes/vertices, in contrast to a graph in which an edge joins two vertices. Formally, a hypergraph is a pair of sets of vertices and hyperedges: $\mathcal{G} \equiv (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$. Since a hypergraph has hyperedges that connect an arbitrary number of nodes, a hyperedge itself is a set of nodes (making $\mathcal{E}_\mathcal{G}$ a set of sets of nodes). Our problem is.

Given: A set of grounded assertions (facts) $\mathcal{F}$, a conjunctive rule/clause $\mathtt{C}$ from a SRL model and a (possibly partial) substitution (instantiations) θ of variables $\mathtt{C}$,

To Do: Return the counts of true groundings $\#(\mathtt{C} \mid \theta)$ of the clause $\mathtt{C}$,

Construct: A partially grounded structural hypergraph motif, $\mathcal{M} \equiv (\mathcal{V}_\mathcal{M}, \mathcal{E}_\mathcal{M})$ representing clause $\mathtt{C}$ and fully grounded hypergraph, $\mathcal{G} \equiv (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ representing grounded assertions ($\mathcal{F}$) - count instances of $\mathcal{M}$ in $\mathcal{G}$ yields an approximation to $\#(\mathtt{C} \mid \theta)$.

Example 4. *A university domain has entity types (variables)* `Professor (p)`, `Student (s)`, `Course (c)`, `Term (t)`, `ResearchProject (r)` *and* `Year (y)`. *We consider two conjunc-*



WorksIn(Amy, MLNs, 2017),
WorksIn(Amy, PRMs, 2017)
WorksIn(Hal, MLNs, 2018)

WorksIn(Ben, PRMs, 2017),
WorksIn(Fei, PRMs, 2017)
WorksIn(Fei, MLNs, 2018),

AdvisedBy(Ben, Amy),
AdvisedBy(Deb, Amy),
AdvisedBy(Fei, Amy),
AdvisedBy(Fei, Hal),

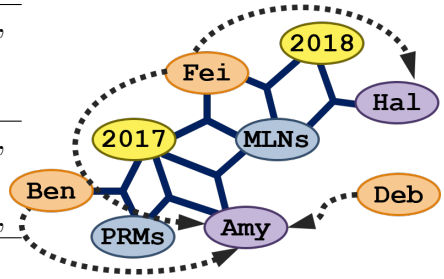


Figure 4.2: (**left**) Motif $\mathcal{M}_2$ for $\mathtt{C}_2$; (**center**) Facts for grounding $\mathcal{M}_2$; (**right**) Ground graph, $\mathcal{G}$. Ternary predicate `WorksIn` are hyperedges in $\mathcal{M}_2$ and $\mathcal{G}_2$. The edges $\mathtt{AdvisedBy}(\mathtt{Deb}, \mathtt{Amy})$ $\mathtt{Deb} \rightarrow \mathtt{Amy}$ and $\mathtt{AdvisedBy}(\mathtt{Fei}, \mathtt{Amy})$ $\mathtt{Fei} \rightarrow \mathtt{Amy}$ also appear in the grounding of $\mathtt{C}_1$ (Fig. 4.1).

tive rules in this domain:

$$\mathtt{AdvisedBy}(\mathtt{s},\mathtt{p}) \wedge \mathtt{Teaches}(\mathtt{p},\mathtt{c},\mathtt{t}) \wedge \mathtt{TA}(\mathtt{s},\mathtt{c},\mathtt{t}) \qquad (\mathtt{C}_1)$$

$$\mathtt{AdvisedBy}(\mathtt{s},\ \mathtt{p}) \ \wedge \ \mathtt{WorksIn}(\mathtt{p},\ \mathtt{r},\ \mathtt{y}) \wedge \mathtt{WorksIn}(\mathtt{s},\ \mathtt{r},\ \mathtt{y}) \qquad (\mathtt{C}_2)$$

The first clause states that `s` *is advised by* `p` *and is a TA for* `c` *that* =`p` *teaches in* `t`. *The second states that* `s` *advised by* `p` *works on* `r` *in a* `y`. *In SRL they are* soft *statements.*

A full grounding refers to a total substitution of values ($\mathbf{t}$) to a set of variables ($\mathbf{v}$), denoted $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$. A partial grounding is an incomplete substitution of values to some variables. With a slight abuse of notation, we do not distinguish between a partial and full grounding, denoting both by θ. A partial grounding will contain a mixture of constants and variables. It must be mentioned that while **we present conjunctive rules from a parameterized logical notation perspective, the same ideas can be easily extended to relational walks/paths in a relational database**. Our approach can easily be extended to clauses of any form by logical transformation. They can be used in discriminative models that use definite clauses (if-then rules), the coverage of the body (a conjunction) of the clause.

Definition 4 (Counts). *For* $\texttt{Relation}(\texttt{v}_1, \ldots, \texttt{v}_t)$*, the predicate counts are the number of true instances of that predicate due to the assertions/facts* $\mathcal{F}$*, given the (partial) grounding* θ *of the its variables. We denote the predicate counts of* $\texttt{Relation}$ $(\texttt{R})$ *as* $n_{\texttt{R}} = \#(\texttt{R} \mid \theta)$.

For a clause $\texttt{C}$*, the clause counts are the number of true instances of* $\texttt{C}$ *in database* $\mathcal{F}$*, given the partial groundings* θ *of the variables in the clause. We denote the clause counts for a clause* $\texttt{C}$ *as* $n_{\texttt{C}} = \#(\texttt{C} \mid \theta)$.

Example 5 (continued). *Consider the facts in Figure 4.1 (center), where* $\texttt{Amy}$ *teaches* 3 *courses* $\{\texttt{AI}, \texttt{ML}, \texttt{Opt}\}$*, teaching* $\texttt{AI}$ *twice in the* $\texttt{Fa17}$ *and* $\texttt{Sp18}$ *terms.* $\texttt{Ben}$*, is a* $\texttt{TA}$ *for* $\texttt{AI}$*, then the count for clause* $\texttt{C}_1$ *given a partial grounding* $\theta_1 = \{\texttt{p}/\texttt{Amy}, \texttt{s}/\texttt{Ben}\}$ *is* $\#(C_1 \mid \theta_1) = 1$*, since* $\texttt{Ben}$ *is a* $\texttt{TA}$ *for only one class. The count for* C_1 *given a partial grounding* $\theta_2 = \{\texttt{P}/\texttt{Amy}, \texttt{T}/\texttt{Fa17}\}$ *is* $\#(C_1 \mid \theta_2) = 1$.

Das et al. (2016) addressed a similar problem but they were restricted to binary (and unary) predicates. But, predicates in $\texttt{C}_1$ and $\texttt{C}_2$ are ternary, an issue that we can handle. Our approach has 3 steps – (i) convert the ground assertions ($\mathcal{F}$) to a hypergraph $\mathcal{G}$, (ii) convert a partially-grounded clause to a partially-grounded structural motif ($\mathcal{M}$), and (iii) count the number of subgraphs matching $\mathcal{M}$ in $\mathcal{G}$.

Definition 5 (Partially Grounded Structural Motif). *A motif* $\mathcal{M}$ *is a Partially Grounded Structural Motif (PGSM) if it is a hypergraph representation of a clause* $\texttt{C}$*, where some of the nodes are parameterized, while others are instantiated to their respective values. That is, a PGSM is a structural motif arising from a partial grounding.*

4.2.1 Conversion to Hypergraphs

Given a clause $\texttt{C}$, we construct a hypergraph motif $\mathcal{M}$ as follows. Each variable in every predicate of $\texttt{C}$ is added as a vertex to $\mathcal{V}_{\mathcal{M}}$, the vertex set of $\mathcal{M}$. Next, all the arguments of a predicate are connected by a hyperedge, which is added to $\mathcal{E}_{\mathcal{M}}$, the edge set of $\mathcal{M}$. Directed edges connect

variables appearing in binary predicates in the order in which they appear, while an n-hyperedge connects the n variables appearing in a n-ary predicate. The nodes of $\mathcal{M}$ correspond to the variables in $\mathtt{C}$ (which can be partially grounded) and the edges correspond to the predicates that contain the respective variables. Given facts ($\mathcal{F}$), a fully-grounded hypergraph $\mathcal{G}$ is similarly constructed. Each constant in $\mathcal{F}$ is added as a vertex to $\mathcal{V}_{\mathcal{G}}$, vertex set of $\mathcal{G}$. Then, all constants appearing in a fact in $\mathcal{F}$ are connected by a hyperedge, and added to $\mathcal{E}_{\mathcal{G}}$, the edge set of $\mathcal{G}$. The construction of $\mathcal{G}$ essentially amounts to parsing assertions (in predicate logic), indexing and insertion into a hypergraph database (Iordanov et al., 2010).

Example 6 (continued). *Figs. 4.1 and 4.2 (left) show motifs $\mathcal{M}_1$ and $\mathcal{M}_2$ for $\mathtt{C}_1$ and $\mathtt{C}_2$ respectively.* `Amy` *advises three students:* `Deb`*, who is a teaching assistant,* `Fei`*, who is a research assistant and* `Ben` *who is both. Thus, $\mathcal{G}_1$, the ground graph for the given assertions (Fig. 4.1, middle & right) Similarly, $\mathcal{G}_2$ (Fig. 4.2) based on the given assertions. Note that,* `Ben` *is both a teaching and a research assistant and appears in both $\mathcal{G}_1$ and $\mathcal{G}_2$. Also, a student may have more than one advisors (Ex:* `Fei` *– 2 advisors:* `Amy` *and* `Hal`*). This highlights various complex interactions among entities and attributes; counting these is critical in inference and learning.*

An important aspect of our work is that, we exploit the notion of *'Partially-Ordered' Hypergraphs* (Feng et al., 2018). In SRL, relations can be interpreted to have an implicit directionality based on the argument order. While translation from argument ordering to normal directed graphs is conceptually straightforward, it is non-trivial in the case of hypergraphs. In directed graphs with loops, unary/binary relations are represented as directed loops/edges from the vertex denoting the first argument to the vertex denoting the second one. Naively, directed hyperedge may be conceived based on the strict ordering of the arguments of an N-ary relation. But, semantically, it is hard to justify. In some domains such strict ordering may make sense while in others it may not.

Example 7. *In the ternary relation '*`teaches(Amy, AI, FA17)`*' the ordering* $\mathtt{professor(p)} \prec \mathtt{course(c)} \prec \mathtt{term(t)}$ *seems most intuitive, considering "*`Amy` *teaches the course* `AI` *in term*

FA17". However, the statement "The course that Amy *teaches in* FA17 *is* AI*" is equally valid, hence the ordering:* p $\prec$ t $\prec$ c*. A relation such as* CoAuthorIn(p, student(s), paper(pa)) *is even more ambiguous. Here the orderings, (1)* p $\prec$ s $\prec$ pa*, (2)* s $\prec$ p $\prec$ pa*, (3)* p $\prec$ pa $\prec$ s *etc., are all semantically similar. Thus strict total ordering is not reasonable.*

A completely undirected representation is also not advantageous since directions allow us to leverage certain properties in our formulation. Thus, we use '*partially-ordered hypergraphs*', where the loops and binary edges have the same protocols as a normal directed graphs. However, for a hyperedge with n nodes $(x_1, \ldots, x_n)$, where the argument order in the original n-ary predicate is $1 \prec 2 \prec \ldots \prec n$, we assume the last node x_n to be the sink node (i.e., hyperedge ends here) and all others as source nodes (hyperedge starts here). It is partial-ordered since ordering exist only between the sink node and a source node and not among the source nodes themselves $(x_{\setminus n} \prec x_n)$. It applies to both the ground hypergraph of assertions as well as the PGSMs.

4.2.2 Approximate Counting via Partially Grounded Structural Motif(s)

We consider the case of counts of conjunctive clauses. Partial grounding in a clause C has 2 scenarios based on number of substitutions. Let number of variables in C be $\ell_{\mathtt{C}}$.

(**Case 1**) When $|\theta| = \ell_{\mathtt{C}}$, that is, when the clause is fully grounded, $\#(\mathtt{C} \mid \theta) = 1$ if $\mathcal{M} \in \mathcal{G}$ else 0. Thus counting, here, is equivalent to checking that the grounded motif $\mathcal{M}$ is a subgraph of $\mathcal{G}$.

(**Case 2**) When $0 \leq |\theta| < \ell_{\mathtt{C}}$, that is, when the clause is either fully lifted ($\ell_{\mathtt{C}} = 0$) or is partially grounded ($|\theta| < \ell_{\mathtt{C}}$), counting is considerably harder. This is the case we address in the rest of this chapter.

Now, for clause C with $\ell_{\mathtt{C}}$ variables, we assume that $m_{\mathtt{C}}$ of these variables are *not grounded*, and $\ell_{\mathtt{C}} - m_{\mathtt{C}}$ variables are grounded. Then, the task is to count the number of groundings of $m_{\mathtt{C}}$, termed as *query* variables, given assignments (θ) to the $\ell_{\mathtt{C}} - m_{\mathtt{C}}$ ground variables. *Ex: For clause* $\mathtt{C}_1$,

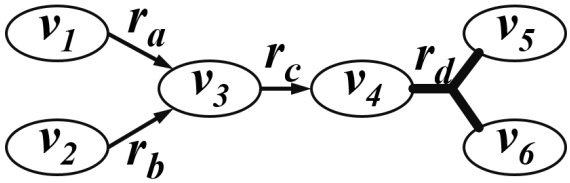


Figure 4.3: The motif $\mathcal{M} \equiv r_a(v_1, v_3) \wedge r_b(v_2, v_3) \wedge r_c(v_3, v_4) \wedge r_d(v_4, v_5, v_6)$. Note that r_d is a hyperedge for the ternary relation $\mathtt{r_d}(\mathtt{v_4}, \mathtt{v_5}, \mathtt{v_6})$.

$\ell_{\mathtt{C_1}} = 4$ *and given a partial grounding* $\theta = \{\, \mathtt{p/Amy},\, \mathtt{s/Ben} \,\}$*, we have* $m_{\mathtt{C_1}} = 2$ *lifted variables (courses* $\mathtt{c}$ *and terms* $\mathtt{t}$*) which we want to count over.*

Without loss of generality, let the first $m_{\mathtt{C}}$ variables in $\mathtt{C}$ be the the *ungrounded* or *query* variables; that is, $v_i,\ i = 1, \ldots, m_{\mathtt{C}}$. Given a motif $\mathcal{M}$ constructed from $\mathtt{C}$, the maximum number of possible subgraphs that match $\mathcal{M}$ in $\mathcal{G}$ is $\prod_{i=1}^{m_{\mathtt{C}}} n_i$, where n_i is the number of possible groundings for *query* variable v_i. Let $P(e \mid \mathcal{G})$ denote the probability of a hyperedge being present in $\mathcal{G}$, then the count of the number of matches of $\mathcal{M}$ in $\mathcal{G}$ is also the clause count, $\#(\mathtt{C} \mid \theta)$:

$$P(\mathcal{M}) \prod_{v \in \mathcal{V}_{\mathcal{M}}} n_v = \left[\prod_{e \in \mathcal{E}_{\mathcal{M}}} P(e \mid \mathcal{G}) \right] \cdot \left[\prod_{v \in \mathcal{V}_{\mathcal{M}}} n_v \right] \tag{1}$$

Intuitively, $P(e \mid \mathcal{G})$ is the *fractional predicate count* that is the ratio of the predicate count given the partial substitutions, to number of groundings of non-substituted variables.

Example 8. *The probability* $P(\mathtt{Teaches(Amy, c, t)} \mid \mathcal{G})$ *can be computed as the number of courses that* $\mathtt{Amy}$ *has taught across all terms divided by the cross-product of the free variables (*$\mathtt{c}$ *and* $\mathtt{t}$*) (total number of possible courses times the number of possible terms). Using the partial substitution* $\theta = \{\, \mathtt{p/Amy} \,\}$ *we have,* $P(\mathtt{Teaches(Amy, c, t)} \mid \mathcal{G}) = \frac{\#(\mathtt{Teaches(p,c,t)} \mid \theta)}{\#(\mathtt{c}) \cdot \#(\mathtt{t})}$*. In Fig. 4.1, this is* $4/(3 \cdot 2) = 2/3$.

Expression (1) presents the expected count. If $P(\mathcal{M})$ is computed exactly, we retrieve the true counts. Since that is intractable we find an approximation.

To understand the intuition behind approximating P, consider the motif in Figure 4.3. The maximum number of times this motif can be present in the ground graph is $\prod_{v \in \mathcal{V}_{\mathcal{M}}} n_v$. The

presence of each hyperedge $e \in \mathcal{G}$ is a Boolean concept (i.e., present or absent in $\mathcal{G}$). Without loss of generality, the joint distribution $P(r_a, r_b, r_c, r_d)$ for Fig 4.3 is,

$$\prod_{e \in \mathcal{M}} P(e \mid \mathcal{G}) = P(r_a) \cdot P(r_b) \cdot P(r_c \mid r_a,\, r_b) \cdot P(r_d \mid r_c).$$

Thus, we now view identifying a motif in a graph as a *search in a directed model with Boolean variables*, where finding an edge depends on the previous edge being found. Note that when any of the edges is absent, the motif will not be present in the grounded graph, and this joint probability is automatically driven to 0. The above expression resembles estimating local models, which is commonly done in standard graphical model estimation. In our case, we further *approximate each of these conditional distributions using summary statistics*.

Reverting to (1), the first term is the product of the individual edge distributions conditioned on the incoming edge, and the second term is the cross-product of the total number of groundings of the query (ungrounded) variables in the motif. The second term is computed a single pre-processing step, followed by caching. For the first term, we employ *summary statistics to approximate this distribution*.

4.2.3 Approximation of clause probability (P)

Summary Computation

In order to approximate the joint distribution P, we employ *graph summaries* similar to the ones used in relational database query engines for cardinality estimation (Schiefer et al., 1998; Neumann and Moerkotte, 2011; Seputis, 2000). Note that these summaries must be selected at the appropriate level of granularity. For instance, finely-grained summaries will correspond to searching the entire database, while highly-aggregated summaries, on the other hand, such as average in-degree and out-degree can lead to poor approximations. Thus, we compute three summaries: (1) node and edge frequencies, (2) node in- and out-degrees, and (3) dependency summaries from hypergraph $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$:

TYPESUM: The **type summary** captures frequencies of node and edge types. Recall that every node in $\mathcal{G}$ corresponds to an entity type, and every edge to a relation type (*edge label* or predicate name). Let $\mathcal{T}$ be the set of entity types, and $\mathcal{R}$ the set of predicate types. With the $\mathsf{TypeOf}(\cdot)$, which returns the node (variable) and edge (relation) types, type summaries $\forall\, \mathtt{T}_i \in \mathcal{T}$ and $\forall\, \mathtt{R}_j \in \mathcal{R}$ are:

$$\#(\mathtt{T}_i) = \#\{\, v \mid v \in \mathcal{V}_\mathcal{G},\, \mathsf{TypeOf}(v) = \mathtt{T}_i \,\}, \tag{2}$$

$$\#(\mathtt{R}_j) = \#\{\, e \mid e \in \mathcal{E}_\mathcal{G},\, \mathsf{TypeOf}(e) = \mathtt{R}_i \,\}. \tag{3}$$

DEGREESUM: **Degree summaries** are the frequencies of incoming and outgoing edges of every node grouped by edge labels. $\mathsf{IncomingEdges}_\mathcal{G}(v)$ and $\mathsf{OutgoingEdges}_\mathcal{G}(v)$ represent the sets of incoming and outgoing edges of v in $\mathcal{G}$. The degree summaries $\forall\, v \in \mathcal{V}_\mathcal{G}$ and $\forall\, \mathtt{R}_j \in \mathcal{R}$ are:

$$\mathsf{In}_\mathcal{G}(v \mid \mathtt{R}_j) = \#\{\, e \mid e \in \mathsf{IncomingEdges}_\mathcal{G}(v),\, \mathsf{TypeOf}(e) = \mathtt{R}_j \,\}, \tag{4}$$

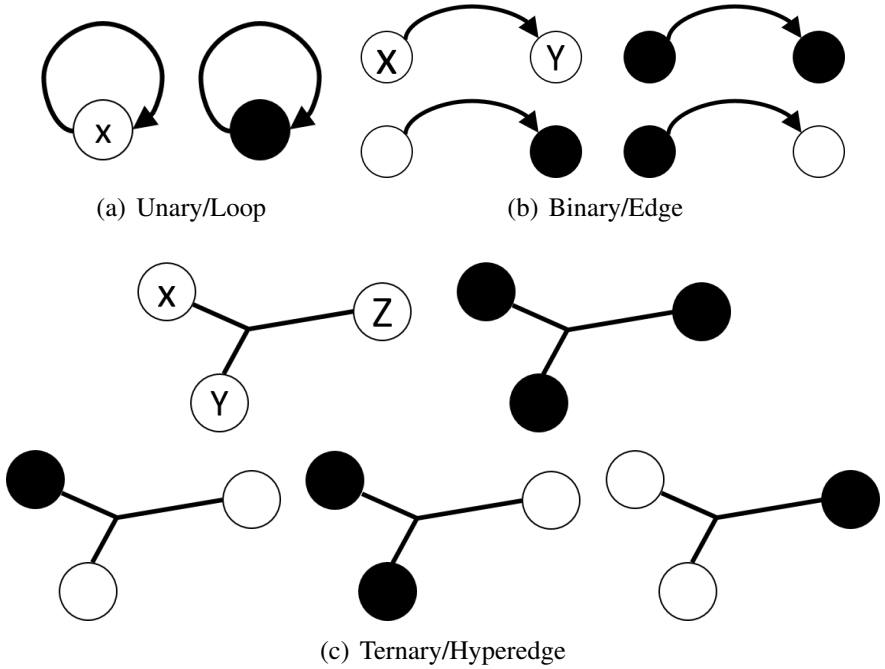
$$\mathsf{Out}_\mathcal{G}(v \mid \mathtt{R}_j) = \#\{\, e \mid e \in \mathsf{OutgoingEdges}_\mathcal{G}(v),\, \mathsf{TypeOf}(e) = \mathtt{R}_j \,\}. \tag{5}$$

DEPENDENCYSUM: Most broadly, we seek to summarize all the **possible dependencies** for a relation $\mathtt{R}_j$: $P(\mathtt{R}_j \mid \mathcal{R} \setminus \mathtt{R}_j)$, that is, the dependency of a R_j on all other relations $\mathcal{R} \setminus \mathtt{R}_j$, which is computationally expensive. Given z relation types in $\mathcal{R}$, we instead construct a $z \times z$ *pairwise dependency matrix*, Δ, whose $(i,\, j)$-th element is $\delta_{ij} = P(\mathtt{R}_i \mid \mathtt{R}_j)$, $\forall\, i,\, j = 1, \ldots, z$. For a pair of relations $\mathtt{R}_i$ and $\mathtt{R}_j$, $P(\mathtt{R}_i \mid \mathtt{R}_j)$ is estimated by sampling paths of length 2 from $\mathtt{R}_j \rightarrow \mathtt{R}_k$. We use these paiwise dependency values for approximation. This matrix is *not symmetric*, since $P(\mathtt{R}_i \mid \mathtt{R}_j)$ will not be the same as $P(\mathtt{R}_j \mid \mathtt{R}_i)$, except under some specific circumstances.

The summary statistics ($\mathbb{S}$) are pre-computed for a ground hypergraph $\mathcal{G}$, and approximate the local distributions:

$$\begin{aligned}\mathbb{S} = [&\{\, \#(\mathtt{T}_i) \}_{\mathtt{T}_i \in \mathcal{T}},\, \#(\mathtt{R}_j) \}_{\mathtt{R}_j \in \mathcal{R}},\\ &\{\, \mathsf{In}_\mathcal{G}(v \mid \mathtt{R}_j) \,\}_{\mathtt{R}_j \in \mathcal{R}},\, \{\, \mathsf{Out}_\mathcal{G}(v \mid \mathtt{R}_j) \,\}_{\mathtt{R}_j \in \mathcal{R}},\\ &\Delta \equiv \{\, P(\mathtt{R}_i \mid \mathtt{R}_j) \,\}_{i,j=1}^{z}].\end{aligned}$$

The computation of $\mathbb{S}$ is performed once as a pre-processing step. The counts for any subsequent query motif $\mathcal{M}$ are computed by estimating local edge distributions using $\mathbb{S}$, without explicitly searching through $\mathcal{G}$. We now explain these cases.



(a) Unary/Loop (b) Binary/Edge

(c) Ternary/Hyperedge

Figure 4.4: Partial grounding in (hyper)edges. Shaded nodes indicate that they are grounded.

Computation of local distributions

The local distribution $P(e)$ of an edge e *with no dependencies* in a motif $\mathcal{M}$, is computed based on grounding of the nodes connected by the edge. For a unary relation or a directed loop such as in Fig. 4.4(a), we have $P(e) = \frac{\#(\mathtt{R}_e)}{\#(\mathtt{T}_x)}$ when fully lifted. Here, $\mathtt{T}_x = \mathsf{TypeOf}(x)$, the entity-class of node x, and

$$P(e \mid \theta_x) = \begin{cases} 1, & \text{if } e \in \mathcal{G}, \text{ when } \theta_x = \{\mathtt{T}_x/x\}, \\ 0, & \text{otherwise.} \end{cases}$$

For the binary case, fully lifted and fully grounded scenarios are computed similarly, with $P(e) = \frac{\#(\mathtt{R}_e)}{\#(\mathtt{T}_x)\cdot\#(\mathtt{T}_y)}$, and

$$P(e \mid \theta_{xy}) = \begin{cases} 1, & \text{if } e \in \mathcal{G}, \text{ when } \theta_{xy} = \{\mathtt{T}_x/x, \mathtt{T}_y/y\}, \\ 0, & \text{otherwise.} \end{cases}$$

Partial groundings, as shown in Fig. 4.4(b) (bottom), are computed via degree summaries:

$$P(e \mid \theta_y) = \frac{\mathsf{In}_{\mathcal{G}}(y \mid e)}{\#(T_x)\cdot 1}, \text{ when } \theta_y = \{\mathtt{T}_y/y\},$$
$$P(e \mid \theta_x) = \frac{\mathsf{Out}_{\mathcal{G}}(y \mid e)}{1\cdot\#(T_y)}, \text{ when } \theta_x = \{\mathtt{T}_x/x\}.$$

While the above distributions can be computed exactly, it is not as straightforward for n-ary relations, which result in hyperedges as shown in Fig. 4.4(c). The fully lifted and fully grounded scenarios are similar to the above cases, but handling partial grounding needs several assumptions:

1. Since, edge directions in an n-hyperedge are not always intuitively clear and may be domain dependent, we follow a partial ordering protocol described earlier.
2. Since for $n > 2$ there can be multiple combinations of partial grounding (shown in Fig. 4.4(c)) and it is intractable to summarize wrt. each, we assume independence among distributions with respect to each grounded node.

When one or more source nodes are grounded $\theta_{x,y} = \{\mathtt{T}_x/x,\, T_y/y\}$ (bottom middle in Fig. 4.4(c))

$$P(e \mid \theta_{x,y}) = \left(\frac{\mathsf{Out}_{\mathcal{G}}(x \mid e)}{1\cdot 1\cdot\#(\mathtt{T}_z)}\right)\cdot\left(\frac{\mathsf{Out}_{\mathcal{G}}(y \mid e)}{1\cdot 1\cdot\#(T_z)}\right).$$

However, if the sink node is grounded $\theta_Z = \{Z/z\}$, as show bottom right in Figure 4.4(c): [$P(e \mid \theta_z) = \mathsf{In}_{\mathcal{G}}(z \mid e)/\left(\#(\mathtt{T}_x)\cdot\#(\mathtt{T}_y)\cdot 1\right)$]. The conditional distribution of an edge preceded by other incoming edges is accessed directly from the conditional dependency matrix, Δ.

Assumptions and Properties

Note that Δ only captures pairwise dependencies and computation of arbitrary dependencies can happen under certain assumptions and properties as outlined below,

Algorithm 4 MACH: **M**otif-based **A**pproximate **C**ounting via **H**ypergraphs

procedure MACH(Clause $\mathtt{C}$, Hypergraph $\mathcal{G}$, Substitution θ, Summary statistics $\mathbb{S}$)
  $\mathcal{M}(\mathcal{V}_{\mathcal{M}}, E_{\mathcal{M}}) \leftarrow$ CONSTRUCTMOTIF($\mathtt{C} \mid \theta$)
  for $v \in \mathcal{V}_{\mathcal{M}}$ **do** ▷ Count node groundings, n_v
    if $v \in \theta$ **then** ▷ v is grounded in θ
      $n_v \leftarrow 1$
    else ▷ v is a variable in θ
      $n_v \leftarrow \#(\mathsf{TypeOf}(v))$ ▷ All values of v
    end if
  end for
  $\chi(\mathcal{M}) \leftarrow \prod_{v \in \mathcal{V}_{\mathcal{M}}} n_v$ ▷ Cartesian product
  $P(\mathcal{M}) \leftarrow$ APPROXJOINT($\mathcal{M}, \theta, \mathcal{G}, \mathbb{S}$) ▷ See Alg 5
  return $\chi(\mathcal{M}) \cdot P(\mathcal{M})$ ▷ As given by Eqn (1)
end procedure

[Independence among incoming edges on same source]. In the motif $\mathcal{M}$, if multiple incoming edges $\mathtt{R}_1, \ldots, \mathtt{R}_j$ are incident on the source node v of the edge $\mathtt{R}_k$, then the dependency $P(\mathtt{R}_k \mid \mathtt{R}_1, \ldots, \mathtt{R}_j)$ factorizes into $\prod_{i=1}^{j} P(\mathtt{R}_k \mid \mathtt{R}_i)$, making the effect of *all preceding edges independent of each other*. This assumption is justified, from a database perspective where *relational joins using the join same attribute are processed independently*.

[Independence due to grounded node]. Two edges in a *PGSM* sharing a common node are rendered independent of each other when the shared node is already instantiated.

[Independence among incoming on different source]. For an n-ary hyperedge, where multiple incoming edges are incident on different source nodes, we assume a similar factorization though the independence property no longer holds, since, constructing summaries for arbitrary dependencies is intractable. While this is a *strong assumption*, in many practical domains, this leads to significant efficiency gains while preserving performance (see experiments).

[Incoming edge to sink is not considered for conditional dependency]. Figure 4.3 clearly shows, that r_b being incoming on the sink node v_3 of r_a $P(r_a \mid r_b)$. However, r_c considers both r_a and r_b as incoming, resulting in $P(r_c \mid r_a,\, r_b)$. Note that, the idea generalizes to any arity.

4.2.4 The MACH Algorithm

Algorithm 4 presents MACH: **M**otif-based **A**pproximate **C**ounting via **H**ypergraphs. As pre-processing, grounded hypergraphs are constructed and summarized (TYPESUM, DEGREESUM, DEPENDENCYSUM). Subsequently, for every clause, a motif is constructed and counted (using the detailed procedure APPROXJOINT presented in Algorithm 5; last page of current chapter).

Algorithm 5 outlines the computation of the joint distribution $P(\mathcal{M})$ for a given PGSM $\mathcal{M}$. The APPROXJOINT procedure accepts a motif $\mathcal{M}$, the partial substitution θ of the motif, the ground hypergraph of assertions/facts $\mathcal{G}$ and the summaries $\mathbb{S}$ constructed in the pre-processing step as the arguments and then iteratively analyzes each (hyper)edge in $\mathcal{M}$ ($\forall e \in \mathcal{E}_{\mathcal{M}}$) **[lines 3-38]**. At any iteration with respect to a given (hyper)edge e, containers $\mathcal{Q}$ and $\mathcal{V}$ store the nodes (of e) which are lifted/query or instantiated/grounded, respectively, based on θ **[lines 4-5]**.

The 3 important cases of partial grounding in a hyperedge e in a motif, namely – (1) fully grounded, (2) fully lifted or (3) partially grounded, are handled explicitly in this procedure.

[CASE 1:] When e is fully grounded (*i.e.* container $\mathcal{Q}$ is empty), probability of edge $P(e)$ is set to $0/1$ based on whether $e|\theta \in \mathcal{E}_{\mathcal{G}}$ **[lines 7-12]**.

[CASE 2:] When e is fully lifted (*i.e.* container $\mathcal{V}$ is empty), $P(e)$ is computed as a ratio of summarized frequency of the relation type R_e and the Cartesian product of the query/lifted nodes in $\mathcal{Q}$, if e has no dependencies.

The frequency estimates are accessible from TYPESUM. In case there are dependencies (incoming edges on nodes of $\mathcal{Q}$) conditionals are accessed from DEPENDENCYSUM $\forall v \in \mathcal{Q}$ and multiplied together[1], $\prod_{f \in \mathcal{E}_{\mathcal{M}} \setminus e} P(\mathtt{R}_e \mid \mathtt{R}_f)$, **[lines 13-18]**. Finally, CASE 3, e being partially grounded ($\mathcal{Q} \neq \emptyset$ & $\mathcal{V} \neq \emptyset$), requires a more involved analysis **[lines 20-35]**. Determining the partial-order of the nodes of e is important when computing probability of the partially-grounded (hyper)edge e. $\mathcal{O}$ and i maintains the index over all *source* nodes and sink node respectively **[line 20]**. First we

[1]Due to properties and assumptions on independence among pairwise conditionals detailed earlier.

inspect all grounded nodes of e (*i.e.* $\mathcal{V}$). Since a grounded node nullifies any possible dependency, we need not worry about conditionals. If a grounded node $v \in \mathcal{V}$ is a source node $v \in \mathcal{O}$, temporary distribution p *w.r.t.* v is given by the ratio of the out-degree of v, $Out_{\mathcal{G}}(v)$, to the Cartesian product of the type frequencies of all the lifted/query nodes. In case of a sink node, $v \in i$, the ratio is with the in-degree of the node, since the (hyper)edge e is assumed to be partially directed from the source nodes to the sink node. Note that v here is grounded, *i.e.* and actual entity is in $\mathcal{G}$ and the degree estimates are accessible from $\mathbb{S}$, specifically DEGREESUM. $P(e)$ is then updated with the product of the temporary distribution of all $v \in \mathcal{V}$ **[lines 21-28]**. Finally, we inspect all lifted/query nodes of e ($q \in \mathcal{Q}$). For any query node q, **iff it is a source node** (refer to properties & assumptions), the temporary distribution p is computed as the product of the conditionals with respect to all incoming edges on q ($\prod_{f \in \mathcal{E}_q} P(\mathtt{R}_e \mid \mathtt{R}_f)$, where $f \in \mathcal{E}_q$ is an incoming edge on q and R_f is its relation type). Again, $P(e)$ is updated with the product of temporary distributions of all $q \in \mathcal{Q}$ **[lines 29-35]**. $P(\mathcal{M})$ is thus the product of all the factors *i.e.* the edge distributions.

Theoretical properties of MACH algorithm

MACH is aimed at efficient and performance-preserving count approximations of satisfied groundings of logical clauses in Statistical Relational models. Efficiency is the key aspect of the approximation strategy. Note, there are two distinct phases in our approach - (1) Pre-processing phase including construction of the ground hypergraph of assertions $\mathcal{G}$ and summary $\mathbb{S}$, and (2) Computing APPROXJOINT phase. While, pre-processing has an asymptotic time complexity of $O(n^2.|R|+n)$, where n is the number of assertions/facts given and $|R|$ is the number of distinct relation types in the domain, it is computed only once for a data set and is thus inconsequential.

Complexity of APPROXJOINT, however, is crucial since it is called arbitrary number of times during structure / parameter learning and inference. Assuming that the summary statistics $\mathbb{S}$ can be accessed efficiently, if we denote the maximum length of a conjunctive clause/motif as k, maximum arity of the clause/motif as $\mathcal{A}$ and maximum in-degree of a node in the motif as d, then the

asymptotic time complexity of APPROXJOINT for any given motif $\mathcal{M}$ is $O(k.\mathcal{A}.d)$. Most SRL systems work with reasonably small clause lengths for tractability, making k a constant. Hence, the effective complexity reduces to $O(\mathcal{A}.d)$. Cartesian product of the nodes in $\mathcal{M}$ ($\prod_{v \in \mathcal{V}_\mathcal{M}} n_v$) is a single operation requiring *constant* time and is inconsequential.

Performance-preserving approximation requires that there should be negligible change (deterioration) in the predictive performance of an SRL model compared to its original performance with exact counts. Our evaluation results in the following section show how MACH adheres to our claim. A *PAC* analysis for approximation error bounds is an interesting future research direction.

4.3 Experimental Evaluation

We investigate the following questions:

(Q1:) Is MACH effective and efficient in full model learning with n-ary relations compared to a robust baseline?

(Q2:) Is modeling n-ary relations faithfully crucial when learning relational model? and

(Q3:) How does MACH compare (scaling vs. performance) to a state-of-the-art database-centric MLN system?

4.3.1 Experimental design

System: We implemented MACH[2] using Java-based *HypergraphDB* architecture (Iordanov et al., 2010), which facilitates construction, operations and persistence of hypergraphs. All database operations related to MACH are done via carefully designed Java methods using the APIs. MACH is a *pluggable* independent module that can be seamlessly integrated with any Java-based SRL system that uses counts.

[2]Code available @ https://github.com/mayukhdas/MACH

Baselines: To evaluate the effectiveness and efficiency of MACH on full structure and parameter learning of MLNs, we compared against the following baselines: (1) *FACT* (Das et al., 2016), which approximates counts via message-passing on normal multi-graphs, integrated with *MLN-Boost* and (2) basic *MLN-Boost*, the vanilla implementation of boosted structure and parameter learner for MLNs (Khot et al., 2011), the state-of-the-art in MLN (full) model learning without count approximations. It is reasonable to expect comparison against scalable learning of MLNs (Venugopal et al., 2015; Sarkhel et al., 2016). However, they cannot handle partial groundings without an exponential blow-up in model / database size since they resort to creating new predicates for every grounding in a clause.

Data Sets: We used three standard SRL data sets: *UW-CSE, Citeseer and WebKB*, a biomedical data set *Carcinogenesis* (Srinivasan et al., 1997), and an NLP/Information-Extraction(IE) data set *NELL-Sports* for evaluation. *UWCSE* is a relation-extraction/link-prediction task over an anonymized representation of staff, students and faculty of 5 different computer science departments; the target is to predict the `AdvisedBy` relation between faculty and students. *Citeseer* is a citation data set for IE, where the target is to predict which field a paper belongs to based on the title. *WebKB* is a consolidated data set of links among the departmental web pages of 4 universities, and the target is to predict if a web page is a faculty page. *Carcinogenesis* is biomedical data set of the structures of chemical compounds (drugs), and the task is to predict if they are carcinogenic. NELL (Carlson et al., 2010) is a system that extracts information from online text, and converts them into a probabilistic knowledge base. *NELL-Sports* is NELL data from the sports domain consisting of information about players and teams. The task is to predict whether a team plays a particular sport.

Measures: We computed AUC-ROC, AUC-PR, CLL, F1 and running times averaged over 5 random train/test splits.

Table 4.1: Results: Performance vs. Efficiency (running time for **L**earning and **I**nference in seconds). ** indicates n-ary predicates.

		Performance				Efficiency	
Data Sets	**Methods**	**AUC–ROC**	**AUC–PR**	**CLL**	**F1**	**L-Time [s]**	**I-Time [s]**
UWCSE**	MACH	0.981	0.337	-0.133	0.217	**13.2**	5.1
	FACT	0.500	0.0068	-0.061	NaN	7.48	2.8
	MLN-Boost	0.998	0.361	-0.134	0.227	27.5	9.4
Citeseer**	MACH	0.998	0.989	-0.173	0.973	**10837**	12.52
	FACT	0.97	0.92	-0.256	0.934	11042	12.45
	MLN-Boost	0.999	0.998	-0.059	0.977	42499	27.42
Carcinogenesis**	MACH	0.525	0.568	-0.811	0.328	108.48	1.8
	FACT	0.500	0.550	-0.704	NaN	102.5	1.5
	MLN-Boost	0.587	0.572	-0.902	0.489	153.84	2.37
WebKB	MACH	1.0	1.0	-0.049	1.0	**5.8**	0.757
	FACT	1.0	1.0	-0.076	1.0	5.97	0.797
	MLN-Boost	1.0	1.0	-0.075	1.0	8.13	0.896
NELL-Sports	MACH	0.78	0.65	-1.43	0.65	253.92	1.03
	FACT	0.76	0.64	-1.38	0.65	238.07	2.01
	MLN-Boost	0.78	0.66	-0.55	0.68	396.24	2.20

4.3.2 Experimental Results

(Q1: full model learning) Table 4.1 summarizes the performance and efficiency results of MACH against the baselines for structure and parameter learning of MLNs. It is clearly evident that there is no real deterioration in predictive performance due to count approximation in MACH (across all data sets) compared to MLN-Boost. However, MACH substantially beats the baselines on efficiency (learning time), especially on larger data sets such as *Citeseer*, where it is about 4 times faster. Even on smaller data sets such as *UW-CSE* and *WebKB*, the difference in learning times, though not as large, is still sizable. This answers **(Q1)** affirmatively.

(Q2: n-ary relations) All data sets except *WebKB* and *NELL-Sports* contain several ternary relations. On such data sets we run FACT by decomposing n-ary relations into $\binom{n}{2}$ binary relations with the order of the arguments aligned with the order in the original relation. We decompose a relation $\mathtt{R}(\mathtt{v}_1, \ldots, \mathtt{v}_n)$, $n > 2$ as $\{\mathtt{R}_{ij}(\mathtt{v}_i, \mathtt{v}_j)\}$, $(i < j)$, that is $i \in [1, n)$, $j \in (i, n]$. We modify the assertions based on the new relations as well. Observe that, though the efficiency gain is equivalent to (and at times better than) MACH, there is a significant deterioration in performance,

Table 4.2: Performance vs. Efficiency (running time for **I**nference) compared against *Tuffy*.

Data Sets	System	AUC-ROC	I-Time [s]
UW-CSE	MACH	0.892	**20.06**
	Tuffy	0.877	37.11
Carcinogenesis	MACH	0.524	**199.8**
	Tuffy	0.560	944.98

especially on the relatively smaller data sets. In the case of a large data set such as *Citeseer*, the performances are relatively similar since the number of spurious assertions introduced by the decomposition of n-ary relations are negligible compared to the sheer size of the original data set. In contrast, this size difference is not prevalent in smaller data sets, which results in worse predictive performance compared to MLN-Boost and MACH. Thus, it is critical that n-ary relations are represented faithfully, which answers **(Q2)** affirmatively.

(Q3: HypergraphDB versus DB) In order to evaluate against a database-centric MLN system, we compared MACH with *Tuffy* (Niu et al., 2011). While *Tuffy* is a robust MLN engine integrated with PostgreSQL RDBMS, an unbiased comparison with our system is challenging. This is because, although *Tuffy* has no restrictions on the type (discriminative vs. generative) of MLNs (unlike MLN-Boost which can only represent discriminative MLNs via Horn clauses), it does not support structure learning (which MLN-Boost can perform, simultaneously with parameter learning). In order to keep the comparison fair, we learn MLNs via MLN-Boost, convert it into *Tuffy* format, and then compare the inference performance and efficiency with MACH. MACH was integrated with the inference engine of MLN-Boost directly. Table 4.2 summarizes the results. Observe how MACH shows significant gain in efficiency, while being at par with *Tuffy*'s performance. Since *Tuffy* reports time inclusive of setup and post-processing, we did the same for MACH. Note that inference time of MACH for *UW-CSE* is 20.06 sec., which is almost twice as fast as *Tuffy*, includes the setup and hypergraph construction and summarization time of 13.26 sec. On *Carcinogenesis* we observe that MACH is approximately 5 times as fast (w/ 192.2 s. setup time). The performances are almost equivalent, though the inference algorithms are different. This

allows us to answer **(Q3)** affirmatively. Finally, note that *Tuffy* infers on all possible instances of the target. For fair comparison we deactivate sub-sampling of negative examples in MLN-Boost.

4.4 Conclusion

We present a generalized method for approximating counts of logical rules in SRL models via transforming both the clauses and the full set of assertions into a hypergraph. We showed how counting the number of subgraphs in these hypergraphs correspond to counting the number of groundings in the SRL model. We presented the use of summary statistics, similar to the idea outlined in Chapter 3, coupled with an intelligent factorization of the motif distribution, to approximate such counts significantly fast.

Algorithm 5 Product of local distributions

procedure APPROXJOINT(Motif $\mathcal{M}$, Substitution θ, Hypergraph $\mathcal{G}$, Summary statistics $\mathbb{S}$)
  $P(\mathcal{M}) \leftarrow 1.0$ ▷ Initialize joint distribution
  for $e \in \mathcal{E}_{\mathcal{M}}$ **do** ▷ For each edge in the motif
    $\mathcal{Q} \leftarrow$ QUERYNODESOF($e \mid \theta$)
    $\mathcal{V} \leftarrow$ GROUNDNODESOF($e \mid \theta$)
    $P(e) \leftarrow 1$ ▷ Initialize edge probability
    if $\mathcal{Q} = \varnothing$ **then** ▷ $e \mid \theta$ is fully grounded
      if $e \in \mathcal{E}_{\mathcal{G}}$ **then**
        $P(e) \leftarrow 1$ ▷ e exists in both $\mathcal{M}$ and $\mathcal{G}$
      else
        $P(e) \leftarrow 0$ ▷ e does not exist in $\mathcal{G}$
      end if
    else if $\mathcal{V} = \varnothing$ **then** ▷ $e \mid \theta$ is fully lifted
      if $\mathcal{E}_{\mathcal{Q}} = \mathsf{IncomingEdges}_{\mathcal{M}}(v) = \varnothing$ **then** ▷ $v \in \mathcal{Q}$
        $P(e) \leftarrow \#(\mathtt{R}_e) / \prod_{v \in \mathcal{Q}} n_v$ ▷ From $\mathbb{S}$
      else
        $P(e) \leftarrow \prod_{\mathcal{Q}} \prod_{f \in \mathcal{E}_{\mathcal{Q}}} P(\mathtt{R}_e \mid \mathtt{R}_f)$ ▷ From $\mathbb{S}$
      end if
    else ▷ $e \mid \theta$ is partially grounded
      $\mathcal{O}, i \leftarrow$ ORDER($\mathcal{V} \cup \mathcal{Q}$, θ)
      for $v \in \mathcal{V}$ **do** ▷ For each grounded node
        if $v \in \mathcal{O}$ **then** ▷ Source nodes, $\mathcal{O}$
          $p \leftarrow \mathsf{Out}_{\mathcal{G}}(v) / \prod_{v \in \mathcal{Q}} n_v$ ▷ From $\mathbb{S}$
        else ▷ Sink node, i
          $p \leftarrow \mathsf{In}_{\mathcal{G}}(v) / \prod_{v \in \mathcal{Q}} n_v$ ▷ From $\mathbb{S}$
        end if
        $P(e) \leftarrow P(e) \cdot p$
      end for
      for $q \in \mathcal{Q}$ **do** ▷ For each query node
        $\mathcal{E}_q \leftarrow \mathsf{IncomingEdges}_{\mathcal{M}}(q)$
        if $\mathcal{E}_q \neq \varnothing$ **and** $q \in \mathcal{O}$ **then** ▷ Source nodes, $\mathcal{O}$
          $p \leftarrow \prod_{f \in \mathcal{E}_q} P(\mathtt{R}_e \mid \mathtt{R}_f)$ ▷ From $\mathbb{S}$
        end if
        $P(e) \leftarrow P(e) \cdot p$
      end for
    end if
    $P(\mathcal{M}) \leftarrow P(\mathcal{M}) \cdot P(e)$
  end for
  return $P(\mathcal{M})$
end procedure

PART II

OBSERVATION QUALITY: DATA-SCARCE ENVIRONMENTS

CHAPTER 5

HUMAN-GUIDED DEEP LEARNING IN SPARSE DATA

Knowledge-augmented learning is a vital component of our envisioned Human-AI collaborative framework, and, that too, across varied levels of representation. For instance, the treatment planning scenario in Example 1 will require the AI agent to learn and reason with low-level signals such as ECG plots or MRI images and so on. Connectionist models, being universal function approximators, are the most likely choices for modeling from such data in low-level representation. Consequently deep learning has re-gained a lot of attention especially for several tasks involving low-level signals. However, as we have noted earlier, faithfully capturing the implicit structure in the domain is equally important. For instance, relationships ECG history of patients sharing kinship etc. Column Networks, a deep architecture, can succinctly capture domain structure and interactions, but may still be prone to sub-optimal learning from sparse and noisy samples. Inspired by the success of human-advice guided learning in AI, especially in data-scarce domains, in this chapter, we present Knowledge-augmented Column Networks (Das et al., 2019), that leverage human advice/knowledge for better learning with noisy/sparse samples.

5.1 Introduction

The re-emergence of Deep Learning (Goodfellow et al., 2016) has found significant and successful applications in difficult real-world domains such as image (Krizhevsky et al., 2012), audio (Lee et al., 2009) and video processing.However, the combinatorial complexity of reasoning in relational domains over a large number of relations and objects has remained a significant bottleneck to overcome. Recent work in relational deep learning has sought to address this particular issue(Kazemi and Poole, 2018; Šourek et al., 2015; Kaur et al., 2017; França et al., 2014). We consider **Column Networks** (CLNs) (Pham et al., 2017) that are composed of several (feedforward) mini-columns each of which represents an entity in the domain. CNs are attractive for a few reasons (1) hidden

layers of a CLN share parameters, which means that making the network deeper does not introduce more parameters, (2) as the depth increases, the CLN can begin to model feature interactions of considerable complexity, which is especially attractive for relational learning, and (3) learning and inference are linear in the size of the network and the number of relations, which makes CLNs highly efficient. However, like other deep learning approaches, CLNs rely on vast amounts of data and incorporate little to no knowledge about the problem domain.

It is well known that biasing learners is necessary in order to allow them to inductively leap from training instances to true generalization over new instances (Mitchell, 1980). While deep learning does incorporate one such bias in the form of domain knowledge (for example, through parameter tying or convolution, which exploits neighborhood information), we are motivated to develop systems that can incorporate richer and more general forms of domain knowledge. One way in which a human can guide learning is by providing *rules over training examples and features*. The earliest such approaches combined explanation-based learning (EBL-NN, (Shavlik and Towell, 1989)) or symbolic domain rules with ANNs (KBANN, (Towell and Shavlik, 1994)). Another natural way a human could guide learning is by expressing *preferences* and has been studied extensively within the preference-elicitation framework due to Boutilier et al. (2006). We are inspired by this form of advice as they have been successful within the context of inverse reinforcement learning (Kunapuli et al., 2013), imitation learning (Odom et al., 2015) and planning (Das et al., 2018a).

These approaches span diverse machine learning formalisms, and they all exhibit the same remarkable behavior: **better generalization with fewer training examples** because they effectively exploit and incorporate domain knowledge as an inductive bias. This is the prevailing motivation for our approach: to develop a framework that **allows a human to guide deep learning** by incorporating rules and constraints that define the domain and its aspects. Incorporation of prior knowledge into deep learning has begun to receive interest recently, for instance, the recent work on incorporating prior knowledge of color and scene information into deep learning for image

classification (Ding et al., 2018). However, in many such approaches, the guidance is not through a human, but rather through a pre-processing algorithm to generate guidance. Our framework is much more general in that a human provides guidance during learning. Furthermore, the human providing the domain advice is not an AI/ML expert but rather a domain expert who provides rules naturally. We exploit the rich representation power of relational methods to capture, represent and incorporate such rules into relational deep learning models. Note that our focus is not combining logic and deep networks as several others have explored this connection for decades since the origin of neuro-symbolic reasoning to more recent ILP-based neural models (Kaur et al., 2017; Kazemi and Poole, 2018) . In our work we use first-order logic as a representation language for human advice and employ it in the context of CLNs.

We make the following contributions: (1) we propose the formalism of Knowledge-augmented Column Networks, (2) we present, inspired by previous work (such as KBANN), an approach to inject generalized domain knowledge in a CLN and develop the learning strategy that exploits this knowledge, and (3) we demonstrate, across four real problems in some of which CLNs have been previously employed, the effectiveness and efficiency of injecting domain knowledge. Specifically, our results across the domains clearly show statistically superior performance with small amounts of data. As far as we are aware, this is the first work on human-guided CLNs.

5.2 Knowledge-augmented Column Networks

Column Networks (Pham et al., 2017) allow for encoding interactions/relations between entities as well as the attributes of such entities in a principled manner without explicit relational feature construction or vector embedding. This enables us to seamlessly transform a multi-relational knowledge graph into a deep architecture making them one of the robust *relational* deep models. Figure 5.1 illustrates an example column network, with respect to the knowledge graph on the left. Note how each entity forms its own column and relations are captured via the sparse inter-column connectors.

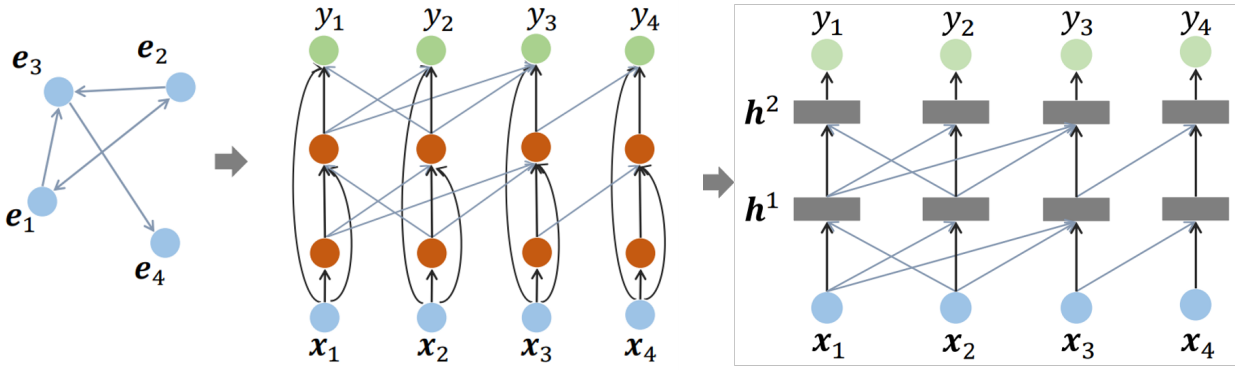


Figure 5.1: Original Column network architecture [diagram source: (Pham et al., 2017)]

Consider a graph $\mathcal{G} = (V, A)$, where $V = \{e_i\}_{i=1}^{|V|}$ is the set of vertices/entities. Without loss of generality, we assume only one entity type. A is the set of arcs/edges between two entities e_i and e_j denoted as $r(e_i, e_j)$. Note that the graph is multi-relational, *i.e.,* $r \in R$ where R is the set of relation types in the domain. To obtain the equivalent Column Network $\mathcal{C}$ from G, let x_i be the feature vector representing the attributes of an entity e_i and y_i its label predicted by the model[1]. h_i^t denotes a hidden node w.r.t. entity e_i at the hidden layer t ($t = 1, \ldots, T$ is the index of the hidden layers). The *context* between 2 consecutive layers captures the dependency of the immediate neighborhood (based on edges/inter-column connectors). For entity e_i, the context w.r.t. r and hidden nodes are computed as,

$$c_{ir}^t = \frac{1}{|\mathcal{N}_r(i)|} \sum_{j \in \mathcal{N}_r(i)} h_j^{t-1}; \tag{5.1}$$

$$h_i^t = g\left(b^t + W^t h_i^{t-1} + \frac{1}{z} \sum_{r \in R} V_r^t c_{ir}^t\right) \tag{5.2}$$

where $\mathcal{N}_r(i)$ are all the neighbors of e_i w.r.t. r in the knowledge graph $\mathcal{G}$. Note the absence of context connectors between h_2^t and h_4^t (Figure 5.1, *right*) since there does not exist any relation between e_2 and e_4 (Figure 5.1, *left*). The activation of the hidden nodes is computed as the sum of the bias, the weighted output of the previous hidden layer and the weighted contexts where

[1]Note that since in our formulation every entity is uniquely indexed by i, we use e_i and i interchangeably

$W^t \in \mathbb{R}^{K^t \times K^{t1}}$ and $V_r^t \in R^{K^t \times K^{t1}}$ are weight parameters and b^t is a bias for some activation function g. z is a pre-defined constant that controls the parameterized contexts from growing too large for complex relations. Setting z to the average number of neighbors of an entity is a reasonable assumption. The final output layer is a softmax over the last hidden layer, $P(y_i = \ell|h_i^T) = softmax\left(b_l + W_l h_i^T\right)$ where $\ell \in L$ is the label (L is the set of labels) and T is the index of the last hidden layer.

Example 9. *Consider a problem of classifying whether a published article is about carcinoid metastasisor is irrelevant, from a citation network, and textual features extracted from the articles themselves. There are several challenges: (1) Data is implicitly sparse due to rarity of studied cases and experimental findings, (2) Some articles may cite other articles related to carcinoid metastasis and contain a subset of the textual features, but address another topic and (3) Finally, the presence of systematic noise, where some important citations were not extracted properly by some citation parser and/or the abstracts are not informative enough.*

The above cases may lead to the model not being able to effectively capture certain dependencies, or converge slower, even if they are captured somewhere in the advanced layers of the deep network. Our approach attempts to alleviate this problem via augmented learning of Column Networks using human advice/knowledge. We formally define our problem in the following manner,

Given: A sparse multi-relational graph $\mathcal{G}$, attributes x_i of each entity (sparse or noisy) in $\mathcal{G}$, equivalent Column-Network $\mathcal{C}$ and access to a Human-expert

To Do: More effective and efficient collective classification by knowledge augmented training of $\mathcal{C}(\theta)$, where $\theta = \langle \{W^t\}_1^T, \{V_r^t\}_{r\in R;t=1}^{t=T}, \{W_\ell\}_{\ell \in L} \rangle$ is the set of all the network parameters of $\mathcal{C}$.

We develop ***K***nowledge-augmented ***C***o***L***umn ***N***etworks (K-CLN), that incorporates human-knowledge, for more effective and efficient learning from relational data (Figure 5.2 illustrates the overall architecture). While knowledge-based connectionist models are not entirely new, our

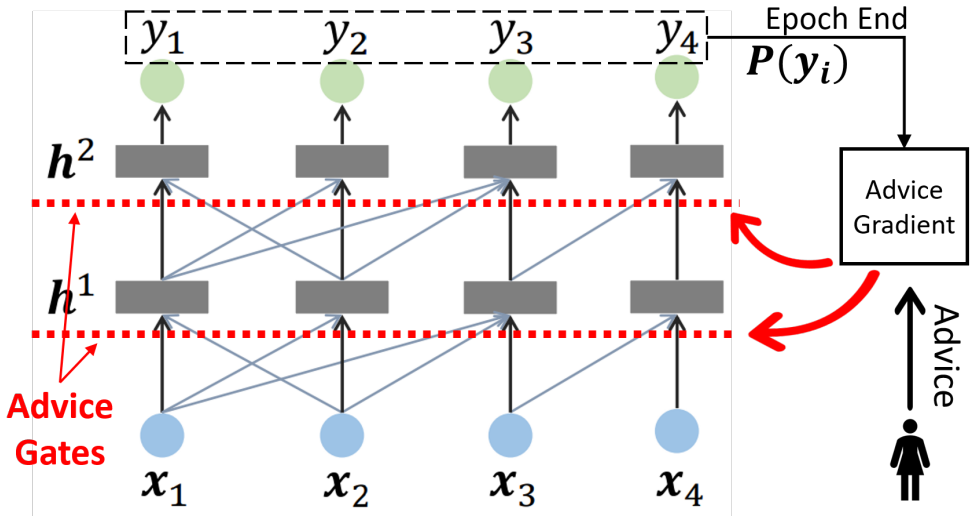


Figure 5.2: K-CLN architecture

formulation provides - (1) a principled approach for incorporating advice specified in an intuitive logic-based encoding/language (2) a deep model for collective classification in relational data.

5.2.1 Knowledge Representation

Any model specific encoding of domain knowledge, such as numeric constraints or modified loss functions etc., has limitations, namely (1) counter-intuitive to the humans since they are domain expert (2) the resulting framework is brittle and not generalizable. Consequently, we employ preferences (akin to IF-THEN statements) to capture human knowledge.

Definition 6. *A preference is a modified Horn clause,* $\wedge_{\mathtt{k},\mathtt{x}} \mathtt{Attr}_{\mathtt{k}}(\mathtt{E}_{\mathtt{x}}) \wedge \ldots \wedge_{\mathtt{r}\in\mathtt{R},\mathtt{x},\mathtt{y}} \mathtt{r}(\mathtt{E}_{\mathtt{x}}, \mathtt{E}_{\mathtt{y}}) \Rightarrow [\mathtt{label}(\mathtt{E}_{\mathtt{z}}, \ell_1) \uparrow; \mathtt{label}(\mathtt{E}_{\mathtt{k}}, \ell_2) \downarrow]$ *where* $\ell_1, \ell_2 \in L$ *and the* $\mathtt{E}_{\mathtt{x}}$ *are variables over entities,* $\mathtt{Attr}_{\mathtt{k}}(\mathtt{E}_{\mathtt{x}})$ *are attributes of* E_x *and* $\mathtt{r}$ *is a relation.* $\uparrow$ *and* $\downarrow$ *indicate the preferred non-preferred labels respectively. Quantification is implicitly* $\forall$ *and hence dropped. We denote a set of preference rules as* $\mathfrak{P}$.

Note that we can always, either have just the preferred label in head of the clause and assume all others as non-preferred, or assume the entire expression as a single literal. Intuitively a rule can

be interpreted as conditional rule, **IF [conditions hold] THEN label ℓ is preferred**. A preference rule can be partially instantiated as well, *i.e.*, or more of the variables may be substituted with constants.

Example 10. *For the prediction task mentioned in Example 9, a possible preference rule could be,*

$$\texttt{hasWord}(\texttt{E}_1, \text{“}\texttt{AI}\text{”}) \wedge \texttt{hasWord}(\texttt{E}_2, \text{“}\texttt{domain}\text{”}) \wedge \texttt{cites}(\texttt{E}_2, \texttt{E}_1) \Rightarrow \texttt{label}(\texttt{E}_2, \text{“}\texttt{irrelevant}\text{”}) \uparrow$$

Intuitively, this rule denotes that an article is not a relevant clinical work to carcinoid metastasis if it cites an 'AI' article and contains the word "domain", since it is likely to be another AI article that uses carcinoid metastatis as an evaluation domain.

5.2.2 Knowledge Injection

Given that knowledge is provided as *partially-instantiated* preference rules $\mathfrak{P}$, more than one entity may satisfy a preference rule. Also, more than one preference rules may be applicable for a single entity. The main intuition is that we aim to consider the error of the trained model w.r.t. both the data and the advice. Consequently, in addition to the *"data gradient"* as with original CLNs, there is a *"advice gradient"*. This gradient acts a feedback to augment the learned weight parameters (both column and context weights) towards the direction of the *advice gradient*. It must be mentioned that not all parameters will be augmented. Only the parameters w.r.t. the entities and relations (contexts) that satisfy $\mathfrak{P}$ should be affected. Let $\mathcal{P}$ be the set of entities and relations that satisfy the set of preference rules $\mathfrak{P}$. The hidden nodes (equation 5.1) can now be expressed as,

$$h_i^t = g\left(b^t + W^t h_i^{t-1} \Gamma_i^{(W)} + \frac{1}{z} \sum_{r \in R} V_r^t c_{ir}^t \Gamma_{ir}^{(c)}\right)$$
$$\text{s.t. } \Gamma_i, \Gamma_{i,r} = \begin{cases} 1 & \text{if } i, r \notin \mathcal{P} \\ \mathcal{F}(\alpha \nabla_i^{\mathfrak{P}}) & \text{if } i, r \in \mathcal{P} \end{cases} \tag{5.3}$$

where $i \in \mathcal{P}$ and $\Gamma_i^{(W)}$ and $\Gamma_{ir}^{(c)}$ are advice-based soft gates with respect to a hidden node and its context respectively. $\mathcal{F}()$ is some gating function, $\nabla_i^{\mathfrak{P}}$ is the *"advice gradient"* and α is the

trade-off parameter explained later. The key aspect of soft gates is that they attempt to enhance or decrease the contribution of particular edges in the column network aligned with the direction of the *"advice gradient"*. We choose the gating function $\mathcal{F}()$ as an exponential [$\mathcal{F}(\alpha\nabla_i^{\mathfrak{P}}) = \exp(\alpha\nabla_i^{\mathfrak{P}})$]. The intuition is that soft gates are natural, as they are multiplicative and a positive gradient will result in $\exp(\alpha\nabla_i^{\mathfrak{P}}) > 1$ increasing the value/contribution of the respective term, while a negative gradient results in $\exp(\alpha\nabla_i^{\mathfrak{P}}) < 1$ pushing them down. We now present the *"advice gradient"* (the gradient with respect to preferred labels).

Proposition 1. *Under the assumption that the loss function with respect to advice / preferred labels is a log-likelihood, of the form* $\mathcal{L}^{\mathfrak{P}} = \log P(y_i^{(\mathfrak{P})}|h_i^T)$*, then the advice gradient is,* $\nabla_i^{\mathfrak{P}} = I(y_i^{(\mathfrak{P})}) - P(y_i)$*, where* $y_i^{(\mathfrak{P})}$ *is the preferred label of entity and* $i \in \mathcal{P}$ *and* I *is an indicator function over the preferred label. For binary classification, the indicator is inconsequential but for multi-class scenarios it is essential (*$I = 1$ *for preferred label* ℓ *and* $I = 0$ *for* $L \setminus \ell$*).*

Since an entity can satisfy multiple advice rules we take the *most* preferred label, *i.e.*, we take the label $y_i^{(\mathcal{P})} = \ell$ to the preferred label if ℓ is given by most of the advice rules that e_j satisfies. In case of conflicting advice (i.e. different labels are equally advised), we simply set the advice label to be the label given by the data, $y_i^{(\mathfrak{P})} = y_i$.

Proof Sketch for Proposition 1 Most advice based learning methods formulate the effect of advice as a constraint on the parameters or a regularization term on the loss function. We consider a regularization term based on the advice loss $\mathcal{L}^{(\mathfrak{P})} = \log P(y_i = y_i^{(\mathfrak{P})}|h_i^T)$ and we know that $P(y_i|h_i^T) = \text{softmax}(b_\ell + W_\ell h_i^T)$. We consider $b_\ell + W_\ell h_i^T = \Psi_{(y_i, h_i^T)}$ in its functional form following prior non-parametric boosting approaches (Odom et al., 2015). Thus $P(y_i = y_i^{(\mathfrak{P})}|h_i^T) = \exp(\Psi_{(y_i^{(\mathfrak{P})}, h_i^T)}) / \sum_{y' \in L} \exp(\Psi_{(y', h_i^T)})$. A functional gradient w.r.t. Ψ of $\mathcal{L}^{(\mathfrak{P})}$ yields,

$$\nabla_i^{\mathfrak{P}} = \frac{\partial \log P(y_i = y_i^{(\mathfrak{P})}|h_i^T)}{\partial \Psi_{(y_i^{(\mathfrak{P})}, h_i^T)}} = I(y_i^{(\mathfrak{P})}) - P(y_i) \tag{5.4}$$

Alternatively, assuming a squared loss such as $(y_i^{(\mathfrak{P})} - P(y_i))^2$, would result in an advice gradient of the form $2(y_i^{(\mathfrak{P})} - P(y_i))(1 - P(y_i))P(y_i)$. We observe that in a functional form the advice gradient is the difference between *the true label distribution and the predicted distribution* (or some function of that difference), *irrespective of the the type of loss* we choose to optimize. As illustrated in the K-CLN architecture, at the end of every epoch of training the *advice gradients* are computed and soft gates are used to augment the value of the hidden units as shown in the main section,

$$\Gamma_i, \Gamma_{i,r} = \begin{cases} 1 & \text{if } i, r \notin \mathcal{P} \\ \mathcal{F}(\alpha \nabla_i^{\mathfrak{P}}) & \text{if } i, r \in \mathcal{P} \end{cases}$$

As illustrated in the K-CLN architecture (Figure 5.2), at the end of every epoch of training the *advice gradients* are computed and soft gates are used to augment the value of the hidden units as shown in Equation 5.3.

Proposition 2. *Given that the loss function $\mathcal{H}_i$ of original CLN is cross-entropy (binary or sparse-categorical for the binary and multi-class prediction cases respectively) and the objective w.r.t. advice is log-likelihood, the functional gradient of the modified objective is,*

$$\begin{aligned} \nabla(\mathcal{H}_i') &= (1 - \alpha)\left(y_i I - P(y_i|h^T)\right) + \alpha\left(I_i^{\mathfrak{P}} - P(y_i^{\mathfrak{P}}|h^T)\right) \\ &= (1 - \alpha)\nabla_i + \alpha\nabla_i^{\mathfrak{P}} \end{aligned} \tag{5.5}$$

where $0 \leq \alpha \leq 1$ is the trade-off parameter between the effect of data and effect of advice, I_i and $I_i^{\mathfrak{P}}$ are the indicator functions on the label w.r.t. the data and the advice respectively and ∇_i and $\nabla_i^{\mathfrak{P}}$ are the gradients, similarly, w.r.t. data and advice respectively (Proof in appendix).

Proof for Proposition 2: The original objective function (*w.r.t.* data) of CLNs is cross-entropy. For clarity, let us consider the binary prediction case, where the objective function is now a binary cross-entropy of the form, $\mathcal{H} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(P(y_i)) + (1 - y_i)\log(1 - P(y_i))$. Ignoring the

summation for brevity, for every entity i, $\mathcal{H}_i = y_i \log(P(y_i)) + (1 - y_i)log(1 - P(y_i))$. Extension to the multi-label prediction case with a sparse categorical cross-entropy is straightforward and is an algebraic manipulation task. Now, from Proposition 1, the loss function *w.r.t.* advice is the log likelihood of the form, $\mathcal{L}^{\mathfrak{P}} = \log P(y_i^{\mathfrak{P}}|h^T)$. Thus the modified objective is expressed as,

$$\mathcal{H}'_i = (1 - \alpha) \left[y_i \log \left(P(y_i) \right) + (1 - y_i) log \left(1 - P(y_i) \right) \right] + \alpha \log(P(y_i^{\mathfrak{P}}))$$

where α is the trade-off parameter. $P(y) = P(y|h^T)$ can be implicitly understood. Now we know from Proposition 1 that the distributions, $P(y_i)$ and $P(y_i^{\mathfrak{P}})$, can be expressed in their functional forms, given that the activation function of the output layer is a *softmax*, as $P(y_i) = \exp\left(\Psi_{(y_i, h_i^T)}\right) / \sum_{y' \in L} \exp\left(\Psi_{(y', h_i^T)}\right)$. Taking the functional (partial) gradients (*w.r.t.* $\Psi_{(y_i, h_i^T)}$ and $\Psi_{(y_i^{\mathfrak{P}}, h_i^T)}$) of the modified objective function (Equation From Equation 5.5), followed by some algebraic manipulation we get,

$$\begin{aligned} \nabla(\mathcal{H}'_i) =& (1 - \alpha)[y_i I_i - y_i P(y_i) - P(x_i) + y_i P(y_i)] + \alpha(I_i^{\mathfrak{P}} - P(y_i^{\mathfrak{P}})) \\ =& (1 - \alpha) \left(y_i I - P(y_i) \right) + \alpha \left(I_i^{\mathfrak{P}} - P(y_i^{\mathfrak{P}}) \right) \qquad \text{(From Equation 5.5)} \end{aligned}$$

Hence, it follows from Proposition 2 that the data and the advice balances the training of the K-CLN network parameters $\theta^{\mathfrak{P}}$ via the trade-off hyperparameter α. When data is noisy (or sparse with negligible examples for a region of the parameter space) the advice (if correct) induces a bias on the output distribution towards the correct label. Even if the advice is incorrect, the network still tries to learn the correct distribution to some extent from the data (if not noisy). The contribution of the effect of data versus the effect of advice will primarily depend on α. If both data and human advice are sub-optimal, correct label distribution is not learnable.

5.2.3 The K-CLN Algorithm

Algorithm 6 outlines all the key steps. KCLN(), the main procedure [**lines: 1-14**], trains a Column Network using both the data (the knowledge graph $\mathcal{G}$) and the human advice (set of preference

Algorithm 6 K-CLN: **K**nowledge-augmented **C**o**L**umn **N**etworks

1: **procedure** KCLN(Knowledge graph $\mathcal{G}$, Column network $\mathcal{C}(\theta)$, Advice $\mathfrak{P}$, Trade-off α)
2: K-CLN $\mathcal{C}^{\mathfrak{P}}(\theta^{\mathfrak{P}}) \leftarrow \mathcal{C}(\theta)$ ▷ modified hidden units Eqn 5.3
3: Initialize $\theta^{\mathfrak{P}} \leftarrow \{0\}$ ▷ initialize parameters of K-CLN
4: $\mathcal{M}^{\mathcal{P}} = \langle \mathcal{M}^{W}, \mathcal{M}^{c}, \mathcal{M}^{label} \rangle \leftarrow$ CREATEMASK($\mathcal{G}, \mathfrak{P}$)
▷ mask $\forall$ entities/relations/labels $\in \mathcal{P}$
5: Initial gradients $\forall i \; \nabla^{\mathfrak{P}}_{i,0} = 0$; $i \in \mathcal{P}$
6: **for** epochs k=1 to convergence **do**
▷ convergence criteria same as original CLN
7: Get advice gradients $\nabla^{\mathfrak{P}}_{i,(k-1)}$ for prev. epoch $k-1$
8: Gates $\Gamma^{\mathfrak{P}}_{i}, \Gamma^{\mathfrak{P}}_{i,r} \leftarrow \exp\left(\alpha \nabla^{\mathfrak{P}}_{i} \times \mathcal{M}^{\mathcal{P}}_{i}\right)$
9: Train $\mathcal{C}^{\mathfrak{P}}$ using Equation 5.3; Update $\theta^{\mathfrak{P}}$
10: Compute $\forall i \; P(y_i)$ from $\mathcal{C}^{\mathfrak{P}}$ ▷ for current epoch k
11: Store $\forall i \; \nabla^{\mathfrak{P}}_{i,k} \leftarrow I(y_i^{(\mathfrak{P})}) - P(y_i)$
▷ Obtain $I(y_i^{(\mathfrak{P})}) \Leftarrow \mathcal{M}^{label}$
12: **end for**
13: **return** K-CLN $C^{\mathfrak{P}}$
14: **end procedure**

15: **procedure** CREATEMASK(Knowledge graph $\mathcal{G}$,Advice $\mathfrak{P}$)
16: $\mathcal{M}^{W}[D \times |O|] \leftarrow \emptyset$
▷ D: feature length; $|O|$: # entities where $\mathcal{G} = (O, R)$
17: $\mathcal{M}^{c}[|O| \times |O|] \leftarrow \emptyset$; $\mathcal{M}^{label}[|O| \times L] \leftarrow \emptyset$
▷ $\mathcal{M}^{W}$: entity; $\mathcal{M}^{c}$: context & $\mathcal{M}^{label}$: label mask
18: **for** each preference $p \in \mathfrak{P}$ **do**
19: **if** $\forall i \in O \wedge \forall r \in R : i$ and r satisfies p **then**
20: $\mathcal{M}^{W}[x, i] \leftarrow 1$ ▷ x is the feature affected by p
21: $\mathcal{M}^{c}[i, j] \leftarrow 1$ ▷ $r = \langle i, j \rangle \in R; j \neq i; j \in O$
22: $\mathcal{M}^{label}[i, \ell] \leftarrow 1$; s.t. LABELOF($i|p$) $= \ell$
23: **end if**
24: **end for**
25: **return** $\langle \mathcal{M}^{W}, \mathcal{M}^{c}, \mathcal{M}^{label} \rangle$
26: **end procedure**

rules $\mathfrak{P}$). It returns a K-CLN $\mathcal{C}^{\mathfrak{P}}$ where $\theta^{\mathfrak{P}}$ are the network parameters, which are initialized to any arbitrary value (0 in our case; [**line: 3**]). Our gating functions are piece-wise/non-smooth and apply only to the subspace entities, features and relations that satisfy the preference rules. So, as a pre-processing step, we create tensor masks that compactly encode such a subspace via the procedure CREATEMASK() [**line: 4**].

The network $\mathcal{C}^{\mathfrak{P}}(\theta^{\mathfrak{P}})$ is then trained through multiple epochs till convergence [**lines: 6-12**]. At the end of every epoch the output probabilities and the gradients are computed and stored in a shared data structure [**line: 11**] to be accessed in the next epoch. Training is largely similar to original CLN with *two key modifications* [**line: 9**] - **(1)** Equation 5.3 is the modified expression for hidden units. **(2)** The data trade-off $1 - \alpha$ augments the original loss and the advice trade-off α, is used to compute the gates. Procedure CREATEMASK() [**lines: 15-27**] constructs the tensor mask(s) over the space of entities, features and relations/contexts that are required to compute the gates (as seen in **line: 8**). There are *3 key components* of the advice mask. They are - **(1)** Entity mask $\mathcal{M}^W$ (#entities $\times$ #features), indicates entities and relevant features are affected by the advice, **(2)** Context mask $\mathcal{M}^c$ (#entities $\times$ #entities), indicates the contexts that are affected (relations are directed, so it is asymmetric), **(3)** Label mask $\mathcal{M}^{label}$, indicates the preferred label of the affected entities, in a one-hot encoding. The masks are iteratively computed for every preference [**lines: 19-25**]. This includes satisfiability checking, $p \in \mathfrak{P}$ [**line: 20**], which is achieved via subgraph matching on the knowledge graph $\mathcal{G}$ (preference rule $\equiv$ subgraph template) (Das et al., 2016, 2019b). The components $\mathcal{M}^W$ and $\mathcal{M}^c$ are used in gate computation in main procedure and $\mathcal{M}^{label}$ is used for the indicator $I_i^{\mathfrak{P}}$ in the advice gradient.

5.3 Experimental Evaluation

We investigate the following questions as part of our experiments, -

(Q1) Can K-CLNs learn efficiently with noisy sparse samples i.e., performance?

(Q2) Can K-CLNs learn effectively with noisy sparse samples i.e., speed of learning?

(Q3) How does quality of advice affect performance of K-CLN i.e., reliance on robust advice?

We compare against the original Column Networks architecture with no advice *(Vanilla CLN indicates the original Column Network architecture (Pham et al., 2017))* as a baseline. Our intention

Table 5.1: Evaluation domains and their properties

Domains	#Entities	#Relations	#Features	Target type
Pubmed	19717	$44,338$	500	Multi-class
Corporate	3119	$\sim 1,000,000$	750	Multi-class
Debates	6662	~ 25000	500	Binary
Disaster	8000	35000	504	Binary

is to show how advice/knowledge can guide model learning towards better predictive performance and efficiency, in the context of collective classification using Column Networks. Also, recall that our problem setting is distinct from most existing noise robust deep learning approaches. Thus, we restrict our comparisons to the original work.

5.3.1 Experimental Setup

System: K-CLN extends original CLN architecture, which uses *Keras* as the functional deep learning API with a *Theano* backend for tensor manipulation. We extend this system to include: (1) advice gradient feedback at the end of every epoch, (2) modified hidden layer computations and (3) a pre-processing wrapper to parse the advice/preference rules and create appropriate tensor masks. Since it is not straightforward to access final layer output probabilities from inside any hidden layer using keras, we use *Callbacks* to write/update the predicted probabilities to a shared data structure at the end of every *epoch*. This data structure is then fed via inputs to the hidden layers. Each mini-column with respect to an entity is a dense network of 10 hidden layers with 40 hidden nodes in each layer (similar to the most effective settings outlined in (Pham et al., 2017)). The *advice masks* encode $\mathcal{P}$, *i.e.*, the set of entities and contexts where the gates are applicable (Algorithm 6).

Domains: We evaluate our approach on **four relational** domains – *Pubmed Diabetes* and *Corporate Messages*, (multi-class), and *Internet Social Debates* and *Social Network Disaster Relevance* (binary). *Pubmed Diabetes*[2] is a citation network for predicting whether a peer-reviewed

[2] `https://linqs.soe.ucsc.edu/data`

article is about *Diabetes Type 1, Type 2 or none*, using textual features (TF-IDF vectors) from 19717 pubmed abstracts and $44,338$ citation relationships among them. Here articles are entities with 500 bag-of-words features (TF-IDF word vectors). *Internet Social Debates*[3] is for predicting stance ('for'/'against') about a debate topic from online posts on social debates. It contains 6662 posts (entities) characterized by TF-IDF vectors and ~ 25000 relations of 2 types, 'sameAuthor' and 'sameThread'. *Corporate Messages*[4] is an intention prediction data set of 3119 flier messages sent by corporate groups in the finance domain, with $1,000,000$ *sameSourceGroup* relations. We predict the intention of the message *(Information, Action or Dialogue)*. Finally, *Social Network Disaster Relevance* is a relevance prediction data set of 8000 *Twitter* posts, curated and annotated by crowd with relevance scores. Besides textual ones, we use score features and $35k$ relations among tweets (*'same author'* and *'same location'*). Table 5.1 outlines their important aspects.

Metrics: Following (Pham et al., 2017), we report macro-F1 and micro-F1 scores for the multi-class problems, and F1 scores and AUC-PR for the binary ones. Macro-F1 computes the F1 score independently for each class and takes the average whereas a micro-F1 aggregates the contributions of all classes to compute the average F1 score. Due to space limitation, we show only the Micro-F1 and the AUC-PR results *(complete set of plots - in supplementary appendix)*. For all experiments we use 10 hidden layers and 40 hidden units per column in each layer. All results are averaged over 5 runs.

Human Advice: K-CLN is designed to handle arbitrarily complex expert advice encoded as preference rules. However, even with some relatively simple preference rules K-CLN is more effective in sparse samples. *Eg: In Pubmed, the most complex preference rule is,* $\texttt{HasWord}(\texttt{e}_1, \text{`}\texttt{fat}') \wedge \texttt{HasWord}(\texttt{e}_1, \text{`}\texttt{obese}') \wedge \texttt{Cites}(\texttt{e}_2, \texttt{e}_1) \Rightarrow \texttt{label}(\texttt{e}_2, \texttt{type}_2) \uparrow$. Note how a simple rule, indicating an article citing another one discussing obesity is likely to be about Type2 diabetes, proved

[3] `http://nldslab.soe.ucsc.edu/iac/v2/`

[4] `https://www.figure-eight.com/data-for-everyone/`

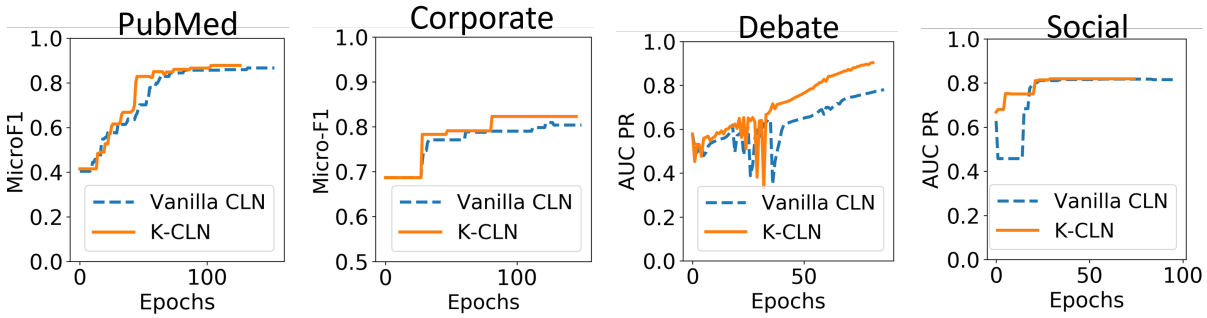


Figure 5.3: Performance w.r.t. **epochs**. *Left to Right* - Pubmed, Corporate Messages, Debates and Social Disaster. *Leftmost 2* show Micro-F1, (multi-class) & *Rightmost 2* show AUC-PR (binary)

to be effective. Knowledge from real physicians can thus, be extremely effective. Sub-optimal advice may lead to a wrong direction of the *Advice Gradient*. The trade-off parameter α balances the effect of advice and data during training.

5.3.2 Evaluation Results

Efficiency (Q1): We present the aforementioned metrics with **varying sample size** and with **increasing epochs** and compare our model against *Vanilla CLN*. We split the data sets into a training set and a hold-out test set with 60%-40% ratio. For varying epochs we only learn on 40% of our pre-split training set (*i.e.*, 24% of the complete data) to train the model and test on the hold-out test set. Figure 5.3 shows that, although both K-CLN and Vanilla CLN converge to the same predictive performance (Micro-F1 for PubMed & Corporate and AUC-PR for the rest), K-CLN converges **significantly faster** (less epochs). Also. for the *corporate* and the *debate*, K-CLN not only **converges faster but also has a better predictive performance** than Vanilla CLN (Figure 5.3- 2^{nd}, 3^{rd}). We have similar observations in Macro-F1 results (see appendix) These results show that K-CLNs learn more *efficiently* with noisy sparse samples thereby answering **(Q1)** affirmatively.

Effectiveness (Q2): The intuition is, *domain knowledge should guide the model to learn better when the training data is sparse*. Thus they are trained on gradually varying sample sizes from 5%

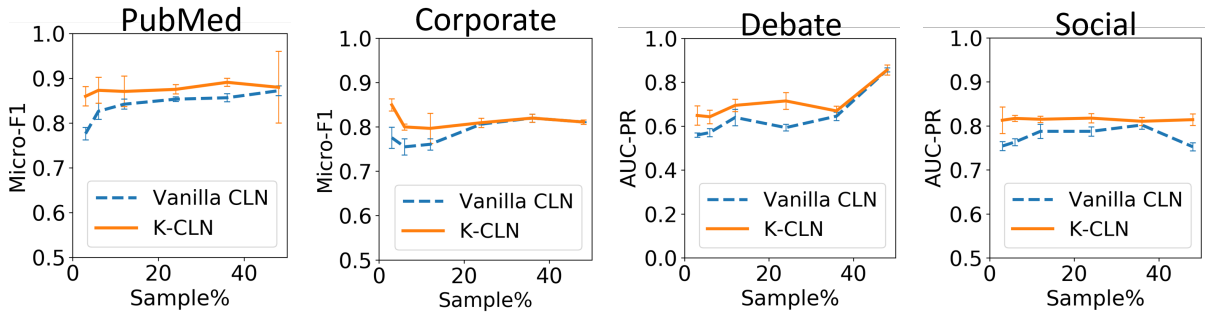


Figure 5.4: Performance w.r.t. **varying samples**. *Left to Right* - Pubmed, Corporate, Debates and Social Disaster. *Leftmost 2* show Micro-F1 & *Rightmost 2* show AUC-PR

of the training data (3% of the complete data) till 80% of the training data (48% of complete data) and tested on the hold-out test set. Figure 5.4 presents the performance results with varying sample sizes for all data sets (again Micro-F1 for PubMed & Corporate and AUC-PR for Internet Debate and Social Disaster). It can be seen that K-CLN outperforms Vanilla CLN across all sample sizes, on both metrics, which suggests that the advice is relevant throughout the training phase with varying sample sizes. For *Corporate Messages*, K-CLN outperforms with small number of samples, gradually converging to a similar prediction performance with larger samples. However, we have observed that, for Macro-F1 (see appendix), the performance is similar for both, although K-CLN performs better with very small samples. This could happen, in multi-class prediction, when the advice may not apply to some of the classes, while it does apply to the rest, effectively averaging out in Macro-F1. Thus K-CLN outperforms the Vanilla CLN, especially in small samples. Thus, K-CLNs learn *effectively* with noisy sparse samples [**(Q2)**].

Robustness (Q3): An obvious question that will arise is – *how robust is our learning system to that of noisy/incorrect advice?* Conversely, *how does the choice of* α *affect the quality of the learned model?* To answer these, we consider the **Internet Social Debates** domain by augmenting the learner with incorrect advice. This incorrect advice is essentially created by changing the preferred label of the advice rules to incorrect values (based on our understating). Also, recall

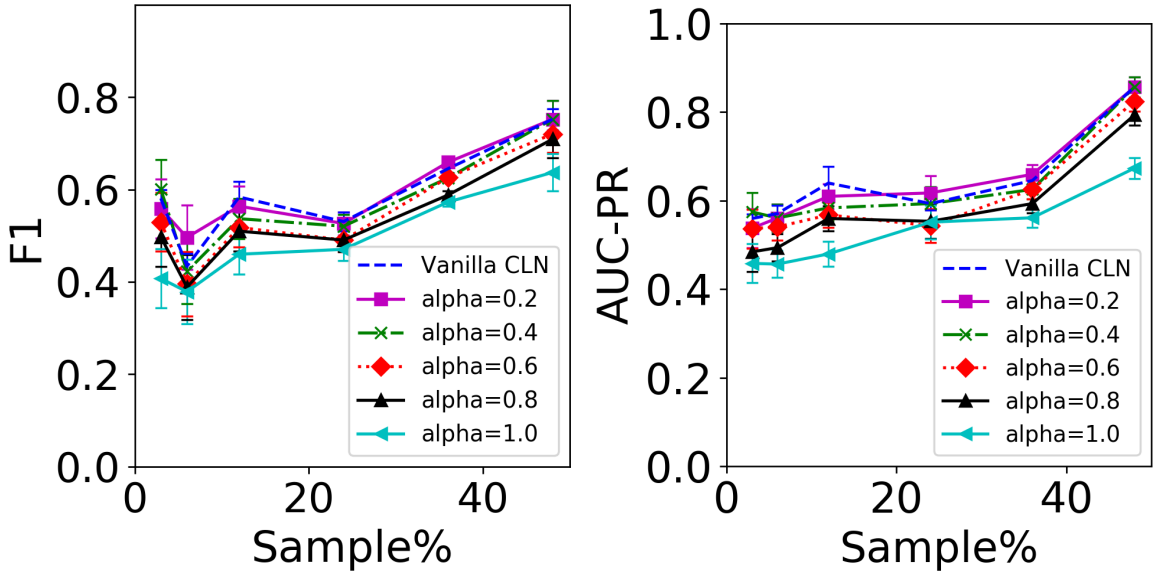


Figure 5.5: Performance, F1 *(Left)* and AUC-PR *(Right)* on **Debates** w/ varying sample sizes & w/ varying ***trade-off parameter*** α (on advice grad.). Note, advice here is incorrect/sub-optimal. $\alpha = 0$ has the identical performance Vanilla CLN [best viewed in color].

that the contribution of advice is dependent on the trade-off parameter α, which controls the robustness of K-CLN to advice quality. Consequently, we experimented with different values of α $(0.2, 0.4, \ldots, 1.0)$, across varying sample sizes. Figure 5.5 shows how with higher α values the performance deteriorates due to the effect of noisy advice. $\alpha = 0$ is not plotted since the performance is same as no-advice/Vanilla CLN. Note that with reasonably low values of $\alpha = 0.2, 0.4$, the performance does not deteriorate and better in some samples. Thus with reasonably low values of α K-CLN is robust to quality of advice **(Q3)**. We picked one domain to present robustness but have observed similar behavior in all the domains.

5.4 Discussion

It is difficult to quantify correctness or quality of human advice unless, absolute ground truth is accessible in some manner. We evaluate on sparse samples of real data sets with no availability of gold standard labels. We have provided the most relevant/useful advice in the experiments aimed

at answering (Q1) and (Q2) as indicated in the experimental setup. We emulate noisy advice (for Q3) by flipping/altering the preferred labels of advice rules in the original set of preferences. We have shown theoretically, in Proposition 1 and 2, that the robustness of K-CLN depends on the advice trade-off parameter α. We illustrated how it can control the contribution of the data versus the advice towards effective training. We postulate that even in presence of noisy advice, the data (if not noisy) is expected to contribute towards effective learning with a weight of $(1 - \alpha)$. Of course, if both the data and advice are noisy the concept is not learnable (as with any algorithm).

The experiments *w.r.t.* **Q3** (Figure 5.5) empirically support our theoretical analysis. We found that when $\alpha \leq 0.5$, K-CLN performs well even with noisy advice. In the earlier experiments where we use potentially good advice, we report the results with $\alpha = 1$, since the advice gradient is piecewise (affects only a subset of entities/relations). So it is reasonable to assign higher weight to the advice and the contribution of the entities and relations/contexts affected by it, given the advice is noise-free. Also, note that the drop in performance towards very low sample sizes (in Figure 5.5) highlights how learning is challenging in the noisy-data and noisy-advice scenario and aligns with our general understanding of most human-in-the-loop/advice-based approaches in AI. Trade-off between data and advice via a weighted combination of both is a well studied solution (Odom and Natarajan, 2018) and, hence, we adapt the same.

CHAPTER 6

GUIDED SEQUENTIAL DECISION MAKING

The primary objective of the AI agent we envisioned in Example 1 (Chapter 1) is to facilitate the decision making process. Thus we aim to develop a Human-AI collaborative sequential decision making framework that is robust to systematic noise. Robust predictive modeling is merely a component of that framework. *For instance, the clinical decision support agent must be able to facilitate with treatment planning as well as healthcare resource planning, which are sequential decision making scenarios. As described in Example 3, a treatment planning scenario for a patient with metastasized carcinoid tumors is challenging, both due to that lack of clinical trials in drug side effect prediction as well as dearth or absence of demonstrations to learn policies from.*

This chapter aims to describe our progress towards a collaborative sequential decision making framework (Das et al., 2018a,b), where we enable the planning (a sequential decison making paradigm) agent to actively seek help from a human collaborator (preferably a domain expert) via a bi-directional interaction and acquire additional knowledge to generate better plans.

6.1 Introduction

Planning under uncertainty has exploited human (domain) expertise in several different directions (Tan and Pearl, 1994; Myers, 1996; Huang et al., 1999; Allen and Ferguson, 2002; Brafman and Chernyavsky, 2005; Sohrabi and McIlraith, 2008). This contrasts with traditional learning techniques that require large amount of labeled data and treat the human as a "mere labeler". One key research thrust in this direction is that of specifying preferences as advice to the planner in order to constrain the search over the space of plans. While successful, most of the preference specification approaches require that the human input be provided in advance before planning commences. There are at least two main issues with this approach: (1) the human sometimes provides the most "obvious" advice that can be potentially inferred by calculating the uncertainty in the plan space, and (2) the planner may not reach the part of the space where the preferences apply.

We propose a framework in which the planner actively solicits preferences as needed. More specifically, our proposed planning approach computes the uncertainty in the plan explicitly and then queries the human expert for advice as needed. This approach not only alleviates the burden, of specifying all the advice upfront, on the human expert but also allows the learning algorithm to focus on the most uncertain regions of the plan space and query accordingly. Thus, it avoids human effort on trivial regions and improves the relevance of the preferences.

We present an algorithm for active preference elicitation in planning called the *Preference-Guided Planner* (PGPLANNER) where the agent treats the human advice as *soft preferences* and solicits these preferences as needed. Such preferences guide the search process towards, potentially, better quality plans.We consider a Hierarchical Task Network (HTN) planner for this task as it allows for seamless natural interaction with humans who solve problems by decomposing them into smaller problems. HTN planners can facilitate humans in providing knowledge at varying levels of generality. We evaluate our algorithm on several standard domains and a novel blocksworld domain against several baselines. Our results show that this collaborative approach allows for more efficient and effective problem solving compared to both standard planning techniques as well as preference-based counterparts with all preferences provided in advance.

Contributions: Our key contributions include: (1) We introduce active preference elicitation for HTN planning; (2) Our framework treats the human input as soft preferences and allows for a trade-off between potentially a sub-optimal expert and a complex plan space; and (3) We evaluate our algorithm on several tasks and demonstrate its efficacy against other baselines. Unlike active advice-seeking (Odom and Natarajan, 2016a), planning does not have training examples from which to generalize. Instead, we select points to query during the planning process based on estimated quality of the available options. We show that even when learning without examples, an active learning framework can guide the learner towards effective and efficient communication with domain experts.

6.2 Active Preference-Guided Planning

Preference-guided planning employs preferences to guide the search through the space of possible plans, sequence of primitive actions. We make use of HTNs to search through these plans by recursively breaking a higher-level task into sub-tasks until every task can be solved using primitive actions. We briefly discuss the HTN planning framework before elaborating our approach further.

HTN Planning An HTN planner (Ghallab et al., 2004; Erol et al., 1994), one of the well-known neo-classical planners, searches for valid plans in the space of task decompositions. It recursively decomposes the current task into sub-tasks based on pre-defined control knowledge, called *methods*, and adds the new sub-tasks into the current set. If a primitive task is solvable by an atomic action, it is removed from the set of tasks; the corresponding action is then added to the plan, and the state is updated. Remaining *non-primitive* tasks are then decomposed further. The resultant network of decompositions is a task network (Ghallab et al., 2004). Formally,

Definition 7. *A task network is directed acyclic graph* $\mathbb{W} = (N, E)$. *A directed edge* $e = \langle \tau, \tau^{(sub)} \rangle$, *where* $\tau, \tau^{(sub)} \in N$ *and* $e \in E$, *is always from a task* τ *to one of its sub-tasks* $\tau^{(sub)}$ *(i.e.* $\tau^{(sub)} \in subtasks(\tau)$*).*

Definition 8. *A method is a tuple* $m_\tau = (\tau, \mathcal{F}(a, s), \{\tau_j\}_{j=1}^{k})$ *where* τ *is the task for which* m_τ *is applicable,* $\mathcal{F}(a, s)$ *ensures* m_τ *is admissible (when* s *satisfies* a*) in current state* s *(*a *is some admissibility criteria) and* $\{\tau_j\}_{j=1}^{k}$ *is the set of sub-tasks to which the task* τ *will be decomposed on application of* m_τ.

Note that, the above definition formalizes admissibility[1] of methods in a generalized fashion, independent of representational syntax. In HTN domain descriptions, however, admissibility of methods are represented as preconditions or conjunctive formulas in predicate logic that the current state 's' must satisfy.

[1]Not to be confused with admissible heuristics.

Definition 9. *A hierarchical task network problem is defined as* $P = (s_0, w_0, O, M)$ *where* s_0 *is the initial state,* w_0 *is the initial task network,* O *is the set of operators (atomic actions) and* M *is the set of decomposition methods.*

As an intuitive example of how HTNs facilitate human experts in specifying knowledge/feedback, consider the problem of building a house. The primary task is "Build House" which can be decomposed into subtasks: *"Build House"* $\rightarrow$ *["Build Foundation", "Build Walls", "Build Roof"]*$_1$. Again the subtask "Build Foundation" can be decomposed further: *"Build Foundation"* $\rightarrow$ *["Dig x feet", "Reinforcement Bars", "Pour Concrete"]*$_2$ (subscripts indicate decomposition level). Methods guide how such tasks need to be decomposed. Clearly, the varying levels of task abstraction allow humans to provide feedback at different levels of generality. For instance, a human could say "Base needs to be deeper than 10 feet" at the *"Dig x ft"* level (level 2) or the human could also say "Foundation must be $1/3^{rd}$ the height of the house" at the higher *"Build Foundation"* task level (level 1). As we explain the formalism and evaluation of our approach through the following sections it will be clearer that the strength of PGPLANNER essentially lies in its ability to identify the most appropriate level of generality for preference elicitation.

Problem Setting Our work differs from prior research on preference-based planning (Sohrabi and McIlraith, 2008; Sohrabi et al., 2009) in two distinct ways: 1) Our preferences are not used to define the best plan. Instead, they guide the decomposition of the network to efficiently find high-quality plans; and 2) We aim to actively acquire preferences as needed during the search process as opposed to requiring the preferences upfront. Our preferences are formally defined as:

Definition 10. *A user-defined preference is a tuple* $\mathfrak{P} = (\wedge f_i, \tau_j, M^+_{\tau_j}, M^-_{\tau_j})$*, where* $\wedge f_i$ *corresponds to conditions of the current state under which the preference should be applied*[2]*,* τ_j *is the relevant task,* $M^+_{\tau_j}$ *is the set of methods (Definition 8 in section 6.2) which are in the user's preferred set and* $M^-_{\tau_j}$ *is the set of methods which are in the user's non-preferred set.*

[2]We use $\wedge f_i$ to denote that this could be a set of multiple conditions

A preference can be considered an IF-THEN rule where $\wedge f_i$ corresponds to the conditions which the current state should satisfy for preference to apply to a particular task, τ_j, and $M^+_{\tau_j}/M^-_{\tau_j}$ represents the method(s) preferred/non-preferred by the user. It is important to note that a preference may be defined for (1) all instances of a particular task ($\wedge f_i = true$), (2) only a single instance of a task or, (3) any level of abstraction in-between.

Our approach uses these preferences to guide the search through the space of possible decompositions in the HTN. Consider the network shown in Figure 6.1. Each node in the network is labeled by the current state - the current configuration of the blocks - and task. For example, the root node represents the task τ_1 of clearing block B in the state where block F is on A, A is on B, etc. Note that we use predicate notation internally to represent the states. The edges in the network represent decompositions and are labeled with the method name. Method m_1 breaks task τ_1 into the operator $PutOnTable(F)$ and, recursively, the task $clear(B)$.

Example 11. *Figure 6.1 represents a preference in Blocks World.*

$$\mathfrak{P} = (Space(Table), Clear(B), \{PutOnTable\}, \{StackonE\})$$

Shaded green areas represent preferred decomposition while shaded red areas represent non-preferred decompositions.

The preference represents the intuition that it is easier to build arbitrary towers when all the blocks are on the table as they can be positioned quickly. As specified, this single preference can apply at multiple points during the search. The effect of the preferences is to update the distribution that increases the probability of the methods M^+ and decreases the probability of the methods M^-. The subtree for M^+ is highlighted in green while the subtree for M^- is shown in red. While a similar preference can be given upfront, our active approach evaluates whether this preference is necessary by estimating the quality of each method. Therefore, our approach increases the value of each preferences by reducing redundancy that may be present in upfront preferences.

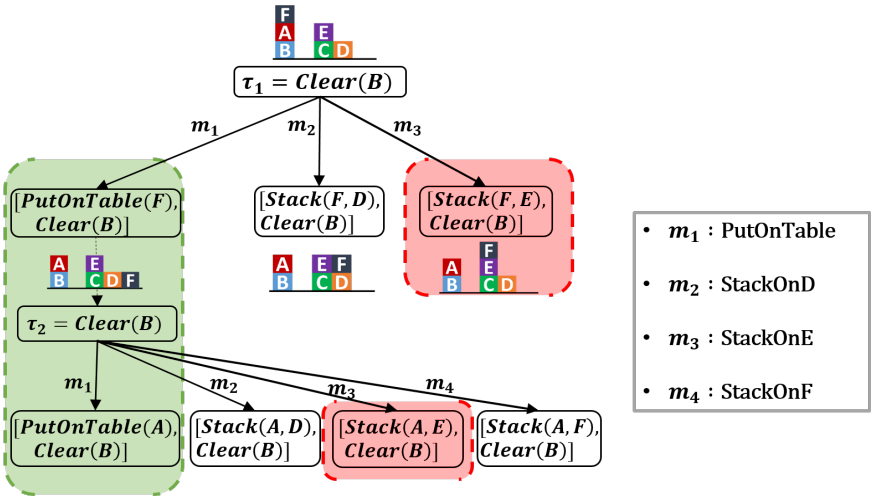


Figure 6.1: Preference guided search in a Blocks world problem. Rectangular nodes signify a set of task(s) to be solved. Admissible methods for decomposing a task τ_1 are m_1, m_2 and m_3. Note, in the lower sub-tree we have an additional admissible method m_4. Block configuration pictures signify current state. The green and red shaded areas denote preferred and non-preferred decompositions (best viewed in color).

In the context of acquiring preferences from the expert, although we assume availability of the expert throughout the planning process, we aim to rely on him/her only when necessary. This setting is similar to stream-based active learning (Freund et al., 1997) where examples are shown online and the algorithm must decide whether to query for a label for the example or ignore it. In contrast to an acquired label, our preferences are more general allowing the expert to prefer/non-prefer methods for decomposition. A query is solicited over the current state s_n and the current task t_n,

Definition 11. *A query is defined over an HTN node n as a tuple $q_n = (s_n, \tau_n)$.*

An expert's response to a query is a preference. As HTNs are hierarchical, the expert is not restricted to providing preferences only over the current state/task. It could also be defined over any subset of the state space that contains s_n. The expert selects the proper generality of the preference. When the space is factored, this involves removing features or introducing variables in description of s_n.

Example 12. *In Figure 6.1,* $Clear(B)$ *at the root has* 3 *different decomposition choices all of which may seem equally valuable to* PGPLANNER *and generate the query:*

$$q = (< Space(Table), On(A, B), On(F, A), ... >, \tau = Clear(B))$$

. The expert may provide the preference specified in Example 11. Note that in that case, the expert gives a general advice that applies to any state in which there is space on the table.

Figure 6.2 illustrates the overall architecture of the PGPLANNER framework. Briefly, at every step (when a task τ needs to be decomposed), PGPLANNER performs a bounded-depth roll-out (simulation of the subsequent decompositions) to evaluate the quality of each method. This allows for estimating a distribution over all methods that can decompose τ ($\mathcal{M}_\tau$). If the uncertainty in this distribution is too high (above ϵ) PGPLANNER decides to seek human guidance via the interface. Human input (preference rule $\mathfrak{P}$) is used to update the distribution. The best method is then selected for proceeding with the decomposition, based on the distribution. The illustration, essentially depicts a snapshot of the decision making pipeline of PGPLANNER. This process occurs at every HTN node that is processed. Subsequent sections elaborate the theory and the functionality of the components of this framework.

6.2.1 Problem Overview

PGPLANNER finds a plan given an HTN problem, defining the initial state/goal task(s), and access to an expert. The goal of PGPLANNER is to find the policy π, a distribution over the methods for each HTN node n, such that the best plan is reached:

$$\arg\min_{\pi} \ (J(\pi) = T\mathbb{E}_{n\sim d_\pi} C_\pi(n)) \tag{6.1}$$

$J(\pi)$ is the total expected cost of finding a plan. If π is used to select decompositions, d_π represents the distribution of HTN nodes reached. T is the depth of the decomposition. $C_\pi(n)$ is the expected cost at node n. $C_\pi(n) = \mathbb{E}_{m\sim\pi_n} C(n, m)$ where $C(n, m)$ is the immediate cost of selecting m at

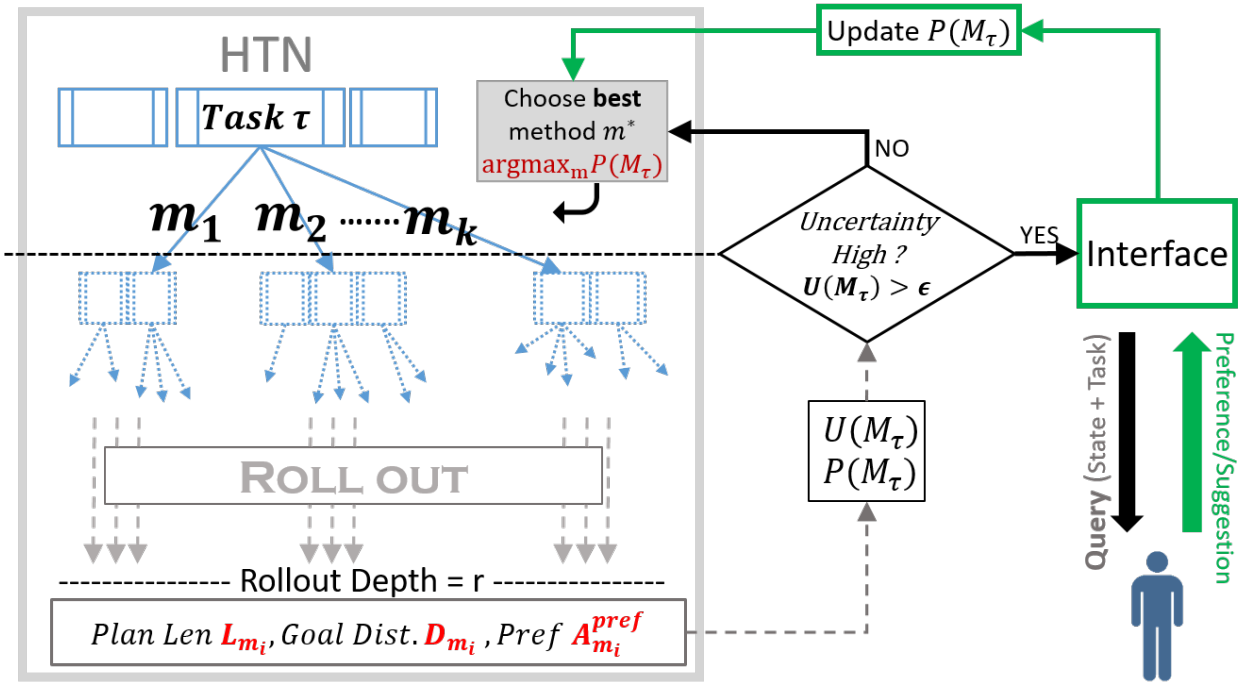


Figure 6.2: Overall architecture of PGPLANNER. At an HTN node, τ represents the task, $M_\tau = \{m_1, m_2, \ldots, m_k\}$ are the methods that can decompose τ. Roll-out allows for estimating distribution over methods and uncertainty $(P(M_{tau}), U(M_\tau))$. Acceptable uncertainty threshold is ϵ. Preference from human expert is used to update the distribution. Method distribution is used to choose the best method.

node n. If the planner aims to find the shortest plan, then the immediate cost C is the number of actions added to the current plan. PGPLANNER's objective is find the best method distribution (with expert guidance) for any task such that the total expected cost is optimized.

PGPLANNER, Algorithm 7 (along with the called procedures 8 & 9), recursively searches through the space of possible HTN decompositions to reach a valid plan. Each node n in the HTN with task τ_n could potentially decompose in several ways according to the available methods (M_{τ_n}). The cost of selecting a method $m \in M_{\tau_n}$ $(C_\pi(n))$ is estimated by rolling out the current plan and then approximating the distance to the goal. The methods are also scored according to the current set of preferences. The overall cost estimate of a method m $(\hat{C}(m))$ is a combination of this preference score and the estimated cost function. Finally, this is converted into a probability

Algorithm 7 Preference-Guided Planning

```
1: procedure PGPLANNER(s_0, w_0, O, M)
2:     Frontier ← all nodes in w_0, Plans = ∅, 𝔓 = ∅        ▷ 𝔓 denotes preference set
3:     while Plans == ∅ and Frontier ≠ ∅ do
4:         currPlan ← RECURSEARCH(∅, Frontier)
5:         if currPlan ≠ ∅ and currPlan ≠ NULL then
6:             Plans ← Plans ∪ currPlan
7:         end if
8:     end while
9:     return Plans
10: end procedure
```

distribution (π) over the methods M_{τ_n}. If this distribution has a high-level of uncertainty (entropy in our case), the expert is queried about the current set of possible methods.

6.2.2 The PGPLANNER Algorithm

The PGPLANNER maintains a $Frontier$, the set of all HTN nodes that have to be explored. It is initialized with 1 or more nodes containing the goal task(s). PGPLANNER proceeds by recursively decomposing the task τ_n of the node n at the head of the frontier and inserting new nodes for the sub-tasks of τ_n. The methods of non-primitive tasks are recursively selected based on $\hat{C}(m)$ (RECURSEARCH, Alg 8, lines **11-20**). Primitive tasks are solved by adding the operator to the current plan (Alg 8, line **22**). This apply step updates the plan for all ancestors of the current node.

When evaluating a node n of the HTN (EVALNODE), the methods $m \in M_{\tau_n}$ represent the set of possible choices. We estimate the cost $\hat{C}(m)$ for each method by rolling out for d steps. L_m represents the estimated cost of the roll-out on method m. In our case it corresponds to the plan length. D_m approximates cost to reach the goal state from the state after the roll-out is completed (EVALNODE, Alg 9, line **4**). This distance, denoted as δ, is the number of unsatisfied goal atoms in the current state. Along with the estimated cost, methods are also evaluated with respect to the set of preferences by the adherence score (A_m). This score (Alg 9, line **7**) is determined by the number of preferences applying to the current node n ($\mathfrak{P}(s_n)$). The number of preferences which

Algorithm 8 Recursive Search

procedure RECURSEARCH($currPlan, Frontier$)
  if $Frontier \neq \emptyset$ **then**
    n = POP($Frontier$)
    if τ_n is *non-primitive* **then**
      $\pi(M_{\tau_n}), U(M_{\tau_n}) \leftarrow$ EVALNODE($n, \mathfrak{P}$)
      if *Not* ACCEPTABLEUNCERTAINTY(U,n) **then**
        $\mathfrak{P} \leftarrow \mathfrak{P} \cup$ QUERYEXPERT(m, s) ▷ Generate query and acquire preference
        $\pi(M_{\tau_n}), U(M_{\tau_n}) \leftarrow$ EVALNODE($n, \mathfrak{P}$)
      end if
      $M_{cur} = M_{\tau_n}$
      while $M_{cur} \neq \emptyset$ **do**
        $M^* \leftarrow \arg\max_m \pi(M_{cur})$ ▷ $m \in M_{curr}$
        $\{n'\} \leftarrow$ DECOMPOSE(τ_n, M^*)
        $NewFrontier \leftarrow$PUSH($Frontier, \{n'\}$) ▷ Temporary frontier stack
        $retVal \leftarrow$ RECURSEARCH($currPlan$, $NewFrontier$) ▷ Recursive call
        if $retVal \neq NULL$ **then**
          return retVal ▷ Backtracking
        end if
        $M_{cur} \leftarrow M_{cur} - M^*$
      end while
    else
      if $Success\,(apply(currPlan, s_n, a_{\tau_n}))$ **then** ▷ Applying primitive action
        return $currPlan$
      end if
    end if
  end if
  return NULL ▷ Backtracking
end procedure

Algorithm 9 Evaluate Node

procedure EVALNODE(HTN node n, Preference $\mathfrak{P}$)
  $\hat{C} = \emptyset$ ▷ Set of scores $\forall m : m \in M_{\tau_n}$
  for *each* $m \in M_{\tau_n}$ **do**
    Node $r_m \leftarrow$ ROLLOUT(m, n, d) ▷ rollout depth d is set to a constant
    $L_m \leftarrow cost(pl_{r_m})$, $D_m \leftarrow \delta(s_{r_m}, goal)$,
    $\mathfrak{P}(s_n) \leftarrow \forall_{p \in \mathfrak{P}}\ (s_n \vDash \wedge f_i^j) \wedge (\tau_p = \tau_n)$ ▷ applicable & state satisfies conditions
    $A_m \leftarrow N_m^+(\mathfrak{P}(s_n)) - N_m^-(\mathfrak{P}(s_n))$
    $\hat{C} \leftarrow \hat{C} \cup \langle m, ((D)^{-1} + (L)^{-1} + A)\rangle$ ▷ Maximize adherence & minimize cost
  end for
  Compute $\pi(M_{\tau_n})$ then $U(M_{\tau_n}) \leftarrow \sum_{m \in M_{\tau_n}} p(m).\log(1/p(m))$ ▷ Entropy from π
  return $\pi(M_{\tau_n}), U(M_{\tau_n})$
end procedure

prefer method m is represented by N_m^+ while N_m^- represents the number of preferences which non-prefer it. Notice that this formulation could allow conflicting preferences on a single method and task. The final score ($\hat{C}(m)$) is a combination of the estimated cost (L_m, D_m) and the adherence score (line **8**). We convert this score into a distribution ($\pi(n)$) over the methods using a Boltzmann softmax, $p(m) = e^{\hat{C}(m)} / \sum_{x \in M_{\tau_n}} e^{\hat{C}(x)}$ where $m \in M_{\tau_n}$ and $p(m) = \pi(n, m)$. This distribution is used in 2 ways: (1) to select the exploration order of the methods and (2) to decide whether a query is necessary.

The query decision is based on the uncertainty over the set of possible methods (RECURSEARCH, Alg 8, lines **6-9**). Inspired by the success of Active Learning(Settles, 2010), we use the uncertainty measure to query the expert. However, one could replace this with any function that needs to be optimized - cost to goal, depth from the start state or a domain specific utility function etc. to name a few. PGPLANNER uses entropy computed from $\pi(n)$ as the measure of uncertainty. Our framework uses a threshold on the entropy, the ***Not*** ACCEPTABLEUNCERTAINTY ($> \epsilon$), to initiate a query. The expert can either provide decomposition preferences over all tasks of a given type, or specify preferences over a single instance of a task, or even provide a partial plan to solve a particular subtask. This establishes an expressive framework for the expert to interact with the planner. The interaction is driven by the planner, allowing it to only ask as and when needed.

Overall, PGPLANNER interacts with the expert to guide the search through the space of possible plans. In an HTN setting, our approach transforms the problem of search of possible plans to sequential decision making problem in the space of possible task decompositions . This facilitates the expert to specify preference in varying levels of generality. The active elicitation formulation ensures that expert knowledge is obtained at the correct and most relevant level of generality allowing our algorithm to learn potentially better plans in a more efficient manner. We now briefly analyze some of the properties of our formulation.

6.2.3 Properties of PGPLANNER

Difference in obtaining preference at various steps

First, we aim to quantify getting preference in earlier steps when compared to getting preference at later steps. Let us denote the probability of choosing a method according to the optimal plan as $p^o(m)$ given by the Boltzmann distribution. Recall that the cost of selecting method m at HTN node n as $C(n, m)$ and the cost of a policy π is $C_\pi(n) = \mathbb{E}_{m\sim\pi_n} C(n, m)$. Now if we use a boolean error function that is set when the method at node n is not chosen according to the policy ($e(n, m) = I(m \neq \pi^*(n))$, then the error of the policy is $e_\pi(n) = \mathbb{E}_{m\sim\pi_n} e(n, m)$.

Our goal is to minimize the total expected cost of a policy π, $J(\pi) = T\mathbb{E}_{n\sim d_\pi} C_\pi(n)$. Ross and Bagnell (Ross and Bagnell, 2010) have shown for any policy π, $J(\pi) \leq J(\pi^*) + kT\bar{\epsilon}$, where π^* is the optimal policy, T is the task horizon, k is the number of steps to the goal, and $\bar{\epsilon} = \frac{1}{T}\sum_i \epsilon_i$, where $\epsilon_i = \mathbb{E}_{n\sim d_{\pi^*}} e_\pi(n_i)$ is the expected error at node i.

A natural question is, does it benefit to ask the query early or should the planning algorithm wait to query the expert. This can be analyzed using the regret framework. Let π_i and π_j denote the policy π when asking the query at steps i and j respectively ($j > i$). Now, rearranging terms, we can show that $J_{\pi_i} - J_{\pi_j} = \Delta(j - i)$, where Δ denotes the expected change in error, if assumed to be the same in both the steps i and j. Thus, the difference between the two choices to solicit preference is linear in the time difference between the two steps, and linear in the change in error in the two steps. Here it is clear that soliciting advice early can reduce the expected total cost.

Benefit of Preference over rollout

We now consider briefly analyzing the value of PGPlanner vs a simple rollout based planning. Let us denote the distribution over methods of optimal policy, PGPlanner, and rollout as π^o, π^A and π^R respectively, where $p^i(m) = \pi^i(n, m)$ where $m \in M_{\tau_n}$ is the posterior of choosing a particular method according to the policy. Suppose we compute the KL divergence of the probability

distribution of methods of PGPlanner and rollout with the optimal distribution,

$$D_R = D_{KL}(\pi^o||\pi^R) = \sum_{m \in M(T)} p^o(m).\ln\left(\frac{p^o(m)}{p^R(m)}\right)$$

$$D_A = D_{KL}(\pi^o||\pi^A) = \sum_{m \in M(T)} p^o(m).\ln\left(\frac{p^o(m)}{p^A(m)}\right)$$

obtaining the difference between D_R & D_A as $D_R - D_A = \sum_{m \in M(T)} p^o(m).log\frac{p^A(m)}{p^R(m)}$, which is simply a weighted sum of the log odds of the probability of choosing a method. It is possible to find the best π^A that maximizes this difference by setting $\sum_m p^A(m) = 1$ as a constraint, but this requires having access to the optimal distribution. Since that is unknown in many cases, one can simply observe that when the preferences drive the distribution over methods towards the optimal one, i.e., choose a method that is close to the optimal, the difference is ≥ 0 indicating that the preference is more useful than the simple rollout. We next show empirically, this is indeed the case in many planning problems.

Performance analysis

While PGPLANNER may have an overhead with respect to space and time, owing to the *roll-out* for evaluating the quality of every method, the incurred cost is *not unbounded* and is practically much lower. The roll-out depth is limited to a pre-defined constant d (which is typically reasonably low), and if the worst-case search depth for the actually planning process is Δ then the roll-out overhead (wrt. time) is bounded at $O(\Delta.d)$. The space overhead is also nominal since all data-structures are re-initialized after each roll-out. *The worst-case space complexity of the recursive plan search in* PGPLANNER *is no worse than the original HTN planning system, SHOP2 (Nau et al., 2003), used as the core planner.*

Most importantly, the overhead cost becomes inconsequential on large problems where the combination of search depth and branching factor is very high. In such cases, actively acquired

preferences are highly likely to guide the PGPLANNER away from deeper subtrees, whenever possible. Naturally, it will depend on the quality of human inputs which we assume to be reasonable. Note that the human is treated as an imperfect teacher and not as an adversary. In large problems, guidance provided by preference results in substantial pruning of the search space and, hence, gain in efficiency and performance. Additionally, as upfront preferences are provided by the expert without visibility into the current state, they may not be as effective as actively acquired ones in pruning out low quality decompositions.

6.3 Experimental Evaluation

Our PGPLANNER is built on top of the SHOP2 (Nau et al., 2003) architecture (called JSHOP), an HTN planner. We have extended the base planner to: (1) perform a roll-out to evaluate all admissible methods for decomposing the task, (2) elicit human feedback/preferences based on our evaluation, and (3) utilize the preferences to guide the search.

6.3.1 Experimental Design

Our experiments aim to answer the following questions:

Q1: Does PGPLANNER generate plans efficiently?

Q2: How effective are the generated plans?

Q3: Does active preference elicitation improve the interaction with the expert?

We compared PGPLANNER against several alternate approaches for preference elicitation including (1) *Upfront Preferences* - where all the preferences are specified before planning, similar to (Sohrabi and McIlraith, 2008; Sohrabi et al., 2009), (2) *Random Query* - selecting whether to query randomly (for each step), and (3) *No Preferences* - planning without preferences. In all

of the experiments, we perform the role of the experts in providing preferences. Performing user studies is an interesting future direction.

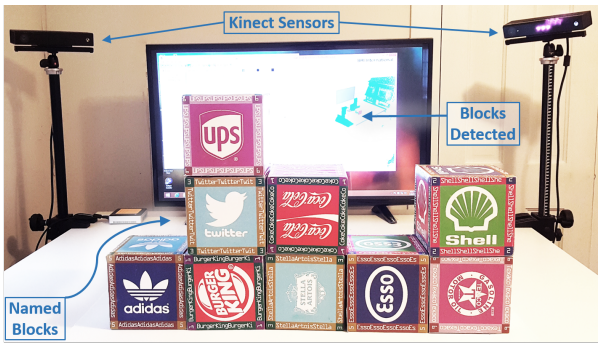


Figure 6.3: Blocks World Apparatus

Table 6.1: Experimental domains

Domains	#[Relations]	#[Objects]	#[Problems]
Freecell	5	52	20
Rovers	27	50	20
Trucks	10	32	20
Depots	6	45	20
Satellite	8	69	20
TidyBot	24	100	10
Towers of Hanoi	10	9	20
Barman	8	50	10
Mystery	12	35	10
Assembly	10	15	10
Rockets	6	15	10
Blocks World	3	40	20

We evaluate PGPLANNER on several standard planning domains as well as a novel Blocks World domain, listed in Table 6.1 (Number of relations, maximum number of objects and number of problems considered in each domain). There is no straightforward way to succinctly describe the complexity of the domains. However, the maximum number of objects and relations in each of them should provide a fair idea of its complexity. Experiments for the Blocks World were performed using a surrogate real-world environment, an apparatus (Figure 6.3) that can detect block configurations of actual named blocks via sensors. State encoding is then generated by processing the sensor data.

For all experiments, we set ACCEPTABLEUNCERTAINTY to $entropy \leq \epsilon \ (= 0.5)$. We also performed line search on the value space, however, a threshold of 0.5 worked well throughout and we report results with that. Preferences were provided by the person designing and conducting the experiments. We avoid experimental bias in the quality of preferences given across all the preference-based approaches. We verify this experimentally by storing all the actively elicited

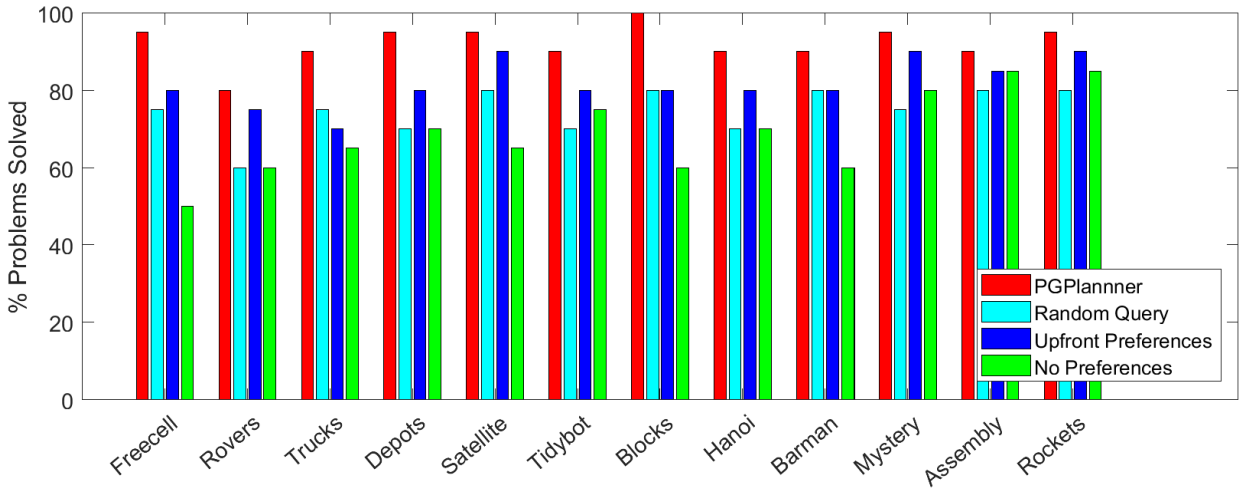


Figure 6.4: Efficiency comparison of all approaches across 12 domains. Percent problems solved in 10 minutes, higher is better. (best viewed in color)

preference statements in a log file and using those as the set of input preferences in the Upfront approach and observing how they affect the decision making process in both cases (see Discussion).

We have developed an interface that facilitates the interaction between the expert and the planner. The interface has three main components: the state module, the partial plan module and the interaction module to visually render the current state, to expose the presently selected set of primitives and to provide a console for the human-agent interaction respectively. The interaction module allows the planner to query the user and the user to respond to the query with a preference.

6.3.2 Experimental Results

In each domain, the planners were executed for 10 minutes. This allows for validating the ability of the expert to guide the algorithm to efficient solutions (as well as accommodating the limited time and attention of the expert). Figure 6.4 shows the percentage of problems where a plan was found. We evaluate the quality of the learned plans separately.

(Efficiency) In most domains, every planner using preferences is able to outperform standard planning (no preferences). This indicates that preferences have a positive impact during planning.

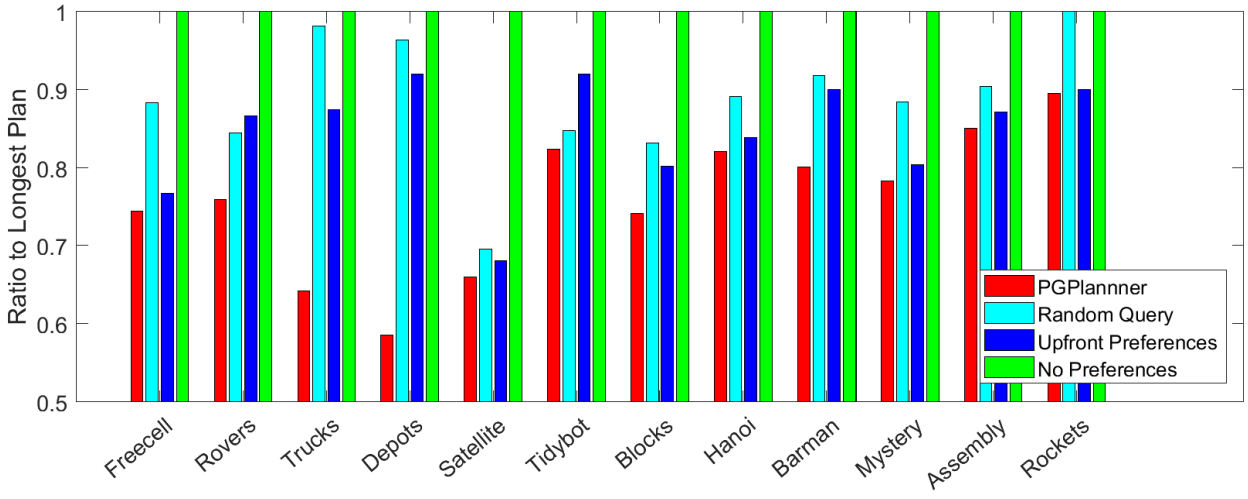
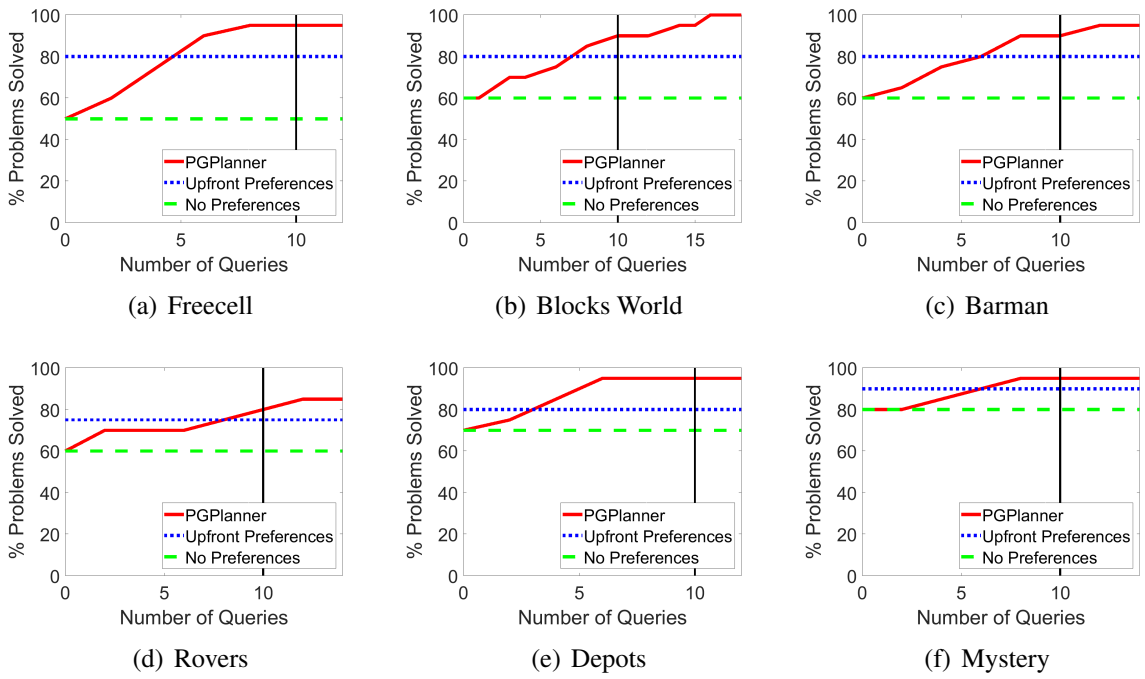


Figure 6.5: Performance comparison of all approaches across 12 domains. Compares the ratio of average plan lengths for every approach to the longest average plan length, lower implies better (best viewed in color)

Planning with upfront preference outperforms randomly querying for preferences in 9 out of 12 domains. A disadvantage of random querying is that it may not query at points in planning where the preferences could have the most impact. Specifying preference upfront can take advantage of preferences at these crucial decisions at the cost of placing additional responsibility on the expert to give useful preferences. Across all domains, PGPLANNER outperforms all of the baselines. This answers **Q1** affirmatively in that actively eliciting preferences guides the planner to solutions more efficiently

(Effectiveness) Next, we investigate the generated plans. In every domain, we only compare problems where all planning methods are able to generate a plan (in the given time constraint). Figure 6.5 illustrates the ratio of the average plan length of each planning method compared to the average of plans generated without preferences[3]. Planning with preferences generates shorter plans in all the domains. Since, as evident, no preference always results in higher average plan lengths. The ratios for all methods with preferences are less than 1. However, PGPLANNER has the lowest

[3]The value for No Preference case is always 1, since the ratio is taken with itself.



(a) Freecell (b) Blocks World (c) Barman

(d) Rovers (e) Depots (f) Mystery

Figure 6.6: Learning Curves in 3 construction (top) and 3 route-finding/ordered-assignment (bottom) domains. Performance: % of problems solved, vs. the # queries. (best viewed in color).

ratio across all domains. Thus, PGPLANNER is able to produce shorter plans on an average than all of the baselines, thus answering **Q2** affirmatively as well.

(Effect of query budget) Finally, we investigate how our method performs relative to the number of queries solicited. Figure 6.6 shows learning curves, in 3 construction oriented domains (Freecell, Blocks World and Barman) and 3 route-finding and ordered-assignment problem solving type domains (Rovers, Depots & Mystery), respectively. Note that learning with no preferences and learning with upfront preferences are constant as the number of preferences never changes. In each domain, the x-axis represents the number of preferences given by the expert. *The vertical line denotes the point in the curve where the number of preferences given upfront equals the number of queries*. Our method outperforms learning with upfront preferences using the same number of preferences in all domains. This suggests that actively eliciting preferences succeeds in generating queries at important stages during the search process, improving the interaction with the expert (af-

firmatively answering **Q3**). Note that, the curves clearly illustrate how the performance varies with the query budget. Interestingly, the rise is gradual. A sharp rise would have indicated that most of the performance gain is achieved via first few preferences towards the early stages of planning (favoring the upfront case). Instead, we observe that PGPLANNER identifies the correct junctures throughout the entire planning process at which expert guidance is most useful, allowing it to pose the optimal set of queries to the human.

6.3.3 Discussion

PGPLANNER is effective, since the framework elicits preferences when and where they are necessary as well as relevant. The expert can provide informative preferences to steer the search process towards more useful parts of the search space. On the other hand, upfront preferences may not always be relevant, because the preferences might be about unreachable regions, or may not alter the decision that the planner would have taken. We verify this by measuring the number of times the preferences were used during the search and how many of those alter the decision of the planner.

Applicability of preferences We observe that on an average, across all domains, upfront preferences are used/applied at least 20% fewer times (corresponding to 2.04 fewer uses) than preferences elicited by PGPLANNER. We evaluated how PGPLANNER queries for (uses) preference across different stages/depths of plan search. Figure 6.7 shows how total average preferences used varies with relative depth. Since planning depth is different for every problem in every domain, 'depth ratio' denotes a standardized scale computed by considering equidistant fractions of the total planing depth. Similarly, as total number of preferences used varies across domains, they were normalized and averaged over all domains, and their cumulative values were plotted. We observe how PGPLANNER acquires 80% of the preferences by 60% of the planning depth as compared to the upfront case which uses preferences uniformly till completion. This empirical result corroborates the earlier discussion on theoretical properties that showed the relationship between impact

of the preference and the relative depth. Note, however, that PGPLANNER does elicit/use some preferences at higher depths. So it cannot be claimed acquiring early would be a valid alternative. It is necessary to identify the right situations where guidance is needed, as done by our approach.

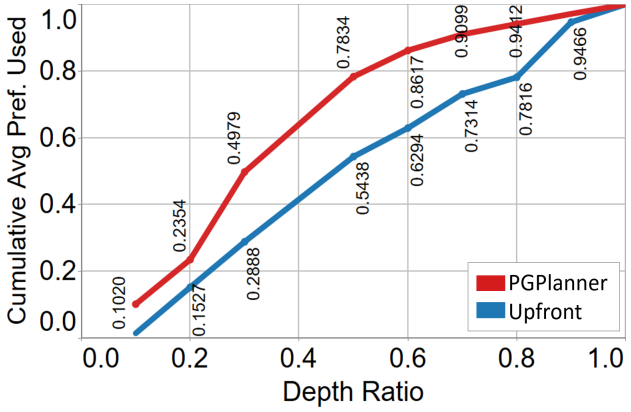


Figure 6.7: Preference used (cumulative) against relative depth (best viewed in color)

Table 6.2: Average % of applicable prefs. that influence decisions.

Domains	PGPLANNER	Upfront Preference
Freecell	91.875	70.625
Rovers	88.125	72.5
Trucks	89	73
Depots	86.25	75.625
Satellite	95.5	84
TidyBot	89	80
Blocks World	89.1	74.28
Towers of Hanoi	84.74	74
Barman	84	80
Mystery	87	86
Assembly	84	83
Rockets	74	73
Average	86.88	77.17

Influence on decision making Furthermore, actively acquired preferences influence the decisions in 86.88% of the cases where the preference applies, compared to 77.17% for upfront preferences. Table 6.2 shows the the statistics of every domain. Notice that, while the measures for planning with upfront preferences are almost always less than PGPLANNER across all domains, the difference is negligible in the 'Mystery' and 'Rockets' domains which aligns with what we observe in terms of performance and efficiency. In the 'Depots' domain, however, the difference in percentage of decision impacting preference is around 11% (row 4) but the difference in average plan length is substantially high (Figure 6.5). On closer inspection we observed that in 2 particular problems in the Depots domain, upfront preferences, though applicable, lead to sub-optimal plans, particularly with substantially high plan length. While it is difficult to discover the exact reason for this behavior, we realized that the 2 particular problems had less 'crates' than 'packages' unlike all other problems which had greater or equal creates as packages. Since this domain essentially

solves ordered-assignment problems, suggestion/preference at the right point is crucial especially in problems where assignment slots are less than the assigned objects, emphasizing the value of our active approach.

We observe exactly the opposite scenario in the 'Towers of Hanoi' domain. Here the difference is around 10%, but we observe that difference in performance (average plan length) is not significantly large. This suggests that *we have not compromised on the quality of preferences provided upfront*. Particularly the upfront preference, *"If possible then* ***avoid empty pegs****"*, seemed to be effective in influencing the planner towards that part of the search space which mostly resulted in better (shorter) plans. A similar, more prominent case, is the 'Freecell' card game domain. The upfront preference *"If possible Then prefer decompositions that allow for immediately finishing a card compared to other choices"* led to shorter plans, even though this preference altered the decisions at only a few points. Clearly, both *Towers of Hanoi* and *Freecell* are intuitive domains and a little practice allows us to formulate high quality upfront preferences. But other domains are more challenging and it is difficult for an expert to imagine all possible scenarios and formulate useful preferences without knowing the current stage and task. Also, planning with upfront preferences performs better than random querying in most domains, indicating that the preferences provided were reasonable. However, PGPLANNER is able to elicit more relevant preferences and use them to find more effective plans efficiently.

Challenges One natural question that arises is that the planner assumes that the human preferences are close to optimal (or at least non sub-optimal). This is indeed a correct observation that is true in many human-in-the-loop systems. In inverse RL (Odom and Natarajan, 2016a) and probabilistic learning (Odom and Natarajan, 2016b), the expert's preferences can be explicitly traded-off with trajectories or labeled data respectively. In such cases, the expert's preferences serve to reduce the effect of targeted noise. However, in planning we do not assume access to such trajectories and instead rely on rollout. Thus, an explicit trade-off is not quite sufficient and warrants a deeper investigation which is beyond the scope of the current work. We note that querying a set of different

experts based on their expertise level on certain sub-tasks remains an interesting and challenging future direction.

6.4 Conclusion

We present a novel method for preference-guided planning where preferences are actively elicited from human experts. PGPLANNER allows for the planner to query only as needed and reduces the burden on the expert to understand the planning process to suggest useful advice. We empirically validate the efficiency and effectiveness of PGPLANNER across several domains and demonstrate that it outperforms the baselines even with fewer preferences. Our formulation transforms plan search into a sequential decision making process in the space of task decompositions, based on estimated distribution and uncertainty. In essence, PGPLANNER is able to infer *what-it-knows* (or does not know) and pose the optimal set of queries to the expert. This not only effectively prunes the sub-optimal regions of the search space but also generates better plans.

CHAPTER 7

CONCEPT INDUCTION FROM SPARSE OBSERVATIONS

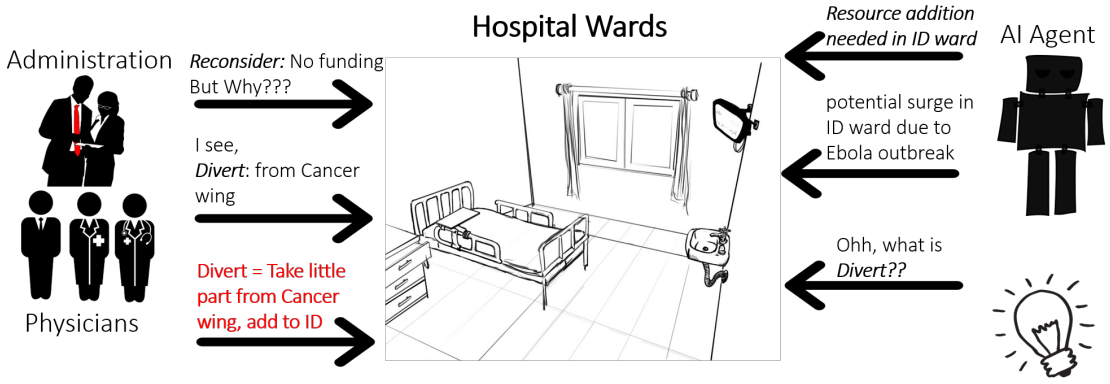


Figure 7.1: Humans teaching new concept *"Divert"* to hospital resource planner agent.

Planning with active preference elicitation, while an effective strategy for an agent to collaboratively acquire additional knowledge, does not address all the challenges of a true human-AI collaborative framework as we envisioned. It is important for the agent to be able to learn novel concepts and progressively enhance its knowledge base (KB) through the course of interaction with human(s). Humans have implicit cognitive and intuitive understanding of varied tangible and intangible concepts. Thus, human collaborators when interacting with the agent will naturally tend to provide additional knowledge in terms of concepts that may not be encoded as part of the agent's KB. Human collaborators would expect the agent to understand that concept seamlessly in subsequent interactions, even if it failed to do so in the beginning. In fact, that is one of the expected outcomes of a Human-AI collaborative framework.

For instance, Figure 7.1 shows an example interaction in the hospital resource planning scenario. Note how the agent is unaware of the concept of divert and the human collaborator provides a single demonstration/instance for that concept of 'divert'. The agent is expected to learn a generalized principle of what 'divert' is, such that it can apply or understand the same in other contexts (such as other wards, other types of resources etc.) as well. In most cases the human collaborator

will provide a single demonstration to convey the idea and the agent should be able to induce a general representation of the concept. In this chapter we present a novel approach to induce concepts in structured spaces from a single (or few) demonstrations *(Under review)*.

7.1 Introduction

We consider the problem of learning generalized representations of concepts using a small number of examples. More specifically, we study the case of learning from *one example*, which is traditionally called *one-shot* learning. This problem has received much attention from ML community (Lake et al., 2011; Khan and Madden, 2014). We consider a challenging setting inside one-shot learning, that of learning explainable and generalizable (first-order) concepts. These concepts can then be particularly reused for learning compositional concepts, for instance, plans. In our concept learning setting, plan induction becomes a special case where a generalizable plan is induced from a single (noise-free) demonstration. As an example, consider building a tower that requires learning *L-shapes* as a primitive. In our formulation, the goal is to learn a *L-shape* from a single demonstration. Subsequently, using this concept, the agent can learn to build a rectangular base (with 2 *L-shapes*) from another single demonstration and so on till the tower is fully built.

Concept learning has been addressed previously in several ways: problem solving by reflection (Stroulia and Goel, 1994), mechanical compositional concepts (Wilson and Latombe, 1994), learning probabilistic programs (Lake et al., 2015), etc. However, all these methods require a significant number of examples. Concept learning as one-class classification has been considered previously where no negative examples are explicitly created. As one would expect, they are considered while learning an SVM or Nearest Neighbor (Tax, 2001), or with a neural network (Kozerawski and Turk, 2018).

While these methods are closely-related, our work has two key differences. First, we aim to learn an *easily interpretable, explainable, and generalizable* concept representation that can then be compounded to learn more complex concepts. To this effect, we focus on learning first-order

horn clause (Horn, 1951)(If Then Else statements). Second, and perhaps most important, we do not assume the existence of a simulator (for plans) or employ a closed-world assumption to generate negative examples. Inspired by Mitchell's observation of futility of bias-free learning (Mitchell, 1997), we develop an approach that employs domain expertise as inductive bias. The principle of structural risk minimization (Vapnik, 1999) shows how optimal generalization from extremely sparse observations is quite hard. The problem is even more critical in structured domains (most relations are false in the world). Thus, one-shot induction of generalized logical concepts is challenging. We aim to solve this problem via an iterative revision of first-order horn clause theories using a novel scoring metric and guidance from a human. *In essence, we emulate a 'student' who learns a generalized concept from an example demonstration provided by the 'teacher', by both reflecting as well as, occasionally, asking relevant questions.*

We propose a novel approach referred as *Guided One-shot Concept Induction* (GOCI) for one-shot concept learning that learns generalized first-order rules. GOCI builds upon an inductive logic program (ILP) learner (Muggleton, 1991) and introduces two key algorithmic steps. First, a modified scoring function that allows for explicitly computing distances between concept representations. We show that this distance corresponds to the well known *Normalized Compression Distance* (NCD) in the case of plan induction. Based on this observation, we demonstrate that the proposed scoring function is indeed a valid distance metric. Second, the use of domain knowledge from human expert as an inductive bias. Unlike many advice taking systems that employ domain knowledge before training, our GOCI algorithm identifies the relevant regions of the concept representation space and *actively* solicits guidance from the human expert to find the target concept in a sample-efficient manner. Overall, these two algorithmic modifications allow for more effective and efficient learning using GOCI that we demonstrate both theoretically and empirically.

Contributions. We make a few important contributions: (1) We derive a new distance-penalized scoring function inside an ILP learner that allows for computing distances between concepts during guided one-shot concept induction. (2) We treat the human-advice as an inductive bias to accelerate

learning. Our ILP learner *actively* solicits richer information from the human experts than mere labels. (3) We analyze the theoretical properties of GOCI. This includes the validity of the distance metric, showing that NCD between plans is a special case of our metric; and deriving a PAC bound based on Kolmogorov complexity. (4) Finally, we demonstrate the exponential gains in both sample efficiency and effectiveness of the algorithm over standard concept learners on diverse concept induction tasks.

7.2 Background discussion

Concept Learning: Concept learning has previously been studied from several perspectives. Our approach is closely related to Stroulia & Goel (1994)'s work which proposes an approach for inducing logical problem-solving concepts by reflection. While our scoring metric in GOCI is similar to 'reflection', its scope is much broader (problem-solving is a special case) and can be used to learn from sparse observations. Wilson (1994) models concepts about continuous geometric shapes by reasoning with two-dimensional bounding boxes. While we use discrete spatial structures as motivating examples, GOCI is not limited to discrete spaces. GOCI is also related in spirit to probabilistic (bayesian) program induction for learning decomposable visual concepts (Lake et al., 2015) which illustrates how exploiting decomposability is more effective than deep learning frameworks. Our approach leverages not only decomposability but implicit relational structure as well, via a logical representation, allowing for generalized concept classes, including plan induction. Tom Mitchell defines concept learning as *inferring a boolean-valued function from training examples of its inputs and outputs* (Mitchell, 1997), which has inspired its treatment as a standard classification/regression problem as well. Our treatment of the problem is slightly different and requires a brief discussion on the notion of concept itself.

Philosophical perspective of concepts: Fodor (1998), holistically outlines the notion of what a 'concept' is and how all other restricted views of the theory of concepts are insufficient. In

summary there is some fundamental desiderata of the notion of a concept as per its cognitive understanding:

P1. Concepts are mental particulars - Any representation of a concept must satisfy the mental ontology of cause and effect.

P2. Concepts are categories - All things in the universe fall into conceptual categories.

P3. Concepts are compositional - Mental representation of concepts derive their contents from the contents of their constituents.

P4. Most concepts need to be learned - only a small fraction of possible concepts exist as innate knowledge that conscious beings are born with.

This work further states how the *'Standard Argument'* of concept acquisition assumes that it is an **inductive process** of creation and testing of candidate hypotheses of a concept (in some representation) against the one or more 'experiences' of that concept.

Our approach is inspired by such a holistic notion of concept and its acquisition. Following (P3) we propose a problem setting for learning decomposabe concepts (formally defined later). Additionally, since some concepts are assumed to be innate knowledge of conscious beings (P4), we emulate that as a pre-defined knowledge-base of known fluents that the AI agent possesses before it starts learning. P2 states the notion of concepts as categories which inspires our treatment of the learning problem as a discriminative one.

As per *'Standard Argument'* concept acquisition is different from the traditional setting of merely learning linear/non-linear decision boundaries in traditional machine learning context, since such learning paradigms fail to align with all the desiderata. Consequently in our learning paradigm, in fact even in ILP, we attempt to induce hypothesis by testing against one or more experiences. The logical structure in our candidate hypotheses is a property of the representational choice and comes as an additional benefit that allows us to adhere to all the desiderata outlined above. Our

problem setting can alternatively be viewed as plan induction, where we induce a plan (ordering is inconsequential) at an higher level abstraction as a representation of the 'mental particular' (P1). In our learning context the experiences we aim to test our candidate hypothesis against, may be very sparse experiences (even a single one).

One/few-Shot Learning: All traditional concept-learning approaches described above require a significant number of examples. Our problem setting requires learning from sparse examples (possibly one). Lake et al., (2011) propose a one-shot version of bayesian program induction of visual concepts. There is also substantial work on one/few-shot learning (both deep and shallow) in a traditional classification setting (Bart and Ullman, 2005; Vinyals et al., 2016; Wang et al., 2018), most of which either pre-train with gold-standard support example set or sample synthetic observations under certain assumptions. We make no such assumptions about synthetic examples.

Theory Induction: Our concept learning approach, aligned with the *Standard Argument* aims to induce and test candidate hypotheses and we choose to represent such hypotheses via logical constructs (aligned with the desiderata of the notion of concepts). ILP (Muggleton, 1991; Muggleton and De Raedt, 1994) inductively learns a logical program (first-order theory) that aims to cover most of the positive examples and none of the negative examples. With ILP, goal is to generalize over instances using background knowledge as search bias by building valid hypotheses about unseen examples. In concept learning, generalization is search through space of candidate inductive hypotheses where induction of single theory requires (1) structuring, (2) searching and (3) constraining space of theories (Lisi, 2008). Research in this area has usually focused on one or more of these dimensions. FOIL (Quinlan, 1990) is an early non-interactive learner with the disadvantage that it occasionally prunes some uncovered hypotheses. This is alleviated in systems like FOCL by introducing `language-bias` in form of user-defined constraints (Pazzani, 1992). Top-down relational ILP systems such as PROGOL (Muggleton, 1995) & ALEPH (Srinivasan, 2007) employ inverse entailment to restrict search space. Later with *Interactive ILP*, learner could

pose questions and elicit expert advice which allows pruning large parts of search space (Sammut and Banerji, 1986; Rouveirol, 1992, 1990). To incorporate new incoming information, ILP systems with *theory revision*, incrementally refine and correct the induced theory (Muggleton, 1988; Sammut and Banerji, 1986). While GOCI is conceptually similar to ALEPH (uses different search strategies and evaluation functions), it additionally acquires domain knowledge by interacting with human expert incrementally. We extend this to one-shot setting. In our setting one-shot necessitates testing candidate hypothesis against one experience, which is extremely difficult. This requires the use of additional knowledge for guiding the induction process towards the optimal generalization.

Knowledge-Guided Learning: Background knowledge in ILP is primarily used as search bias. Strong inductive biases in the form of domain knowledge are required to accelerate learning. Mitchell (1980) proved that biasing learners is necessary to achieve true generalization over new instances. Although earliest form of knowledge injection can be found in explanation-based approaches (Shavlik and Towell, 1989; Towell and Shavlik, 1994), our work relates to preference-elicitation framework (Braziunas and Boutilier, 2006) which guides learning via human preferences. Augmented learning with domain knowledge as an inductive bias has long been explored across various modeling formalisms, including SVMs, ANNs (Fung et al., 2002; Towell and Shavlik, 1994), probabilistic logic (Odom et al., 2015), and planning (Das et al., 2018a,b). Our human-guided GOCI learner aims to extend these directions in the context of learning complex concepts (including plans).

7.3 Guided One-shot Concept Induction

We are inspired by a teacher (human) and student (machine) setting in which a *small number of demonstrations are used to learn generalized concepts* (Chick, 2007). Intuitively, the description provided by a human teacher tends to be modular (can have distinct logical partitions), structured

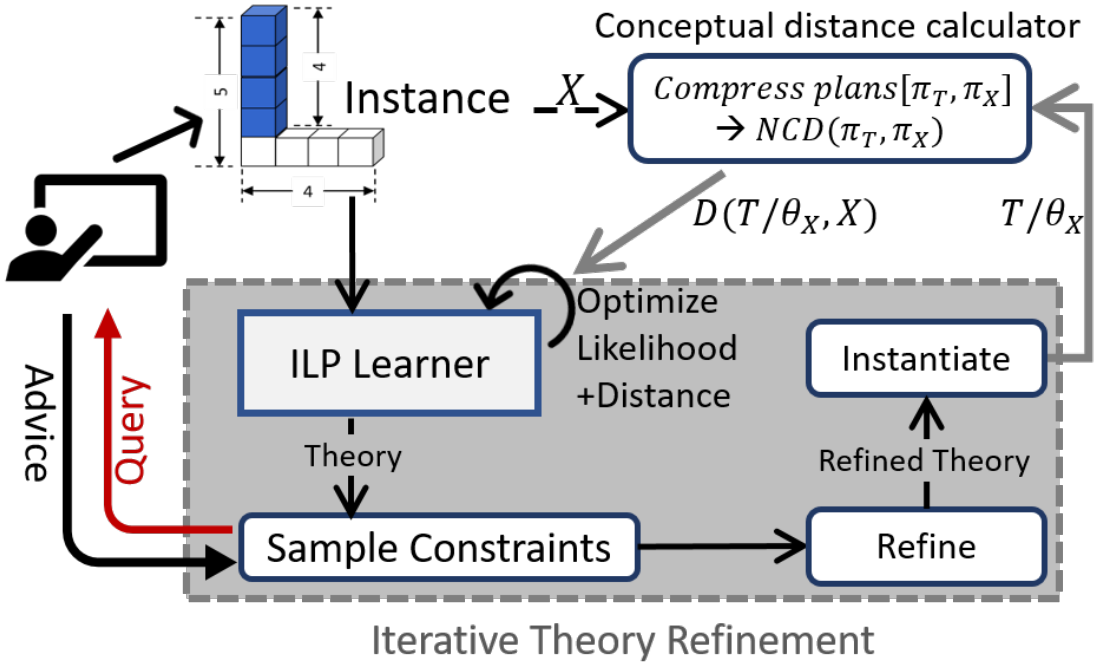


Figure 7.2: Highlevel overview of our GOCI framework.

(entities and relations between them), and in terms of known concepts. Hence, a vectorized representation of examples is insufficient. We choose a logical representation, specifically a *function-free restricted form of first-order logic (FOL)* that models structured spaces faithfully.

Our approach GOCI (Figure 7.2) consists of an ILP learner that induces horn clauses from data. Typical ILP learners use coverage to score candidate clauses. However, this cannot be used with one-shot learning. Hence, we introduce a penalty function. Specifically, this penalty function is the conceptual distance between the induced theory at the current iteration and the demonstration provided by the "teacher" (*Conceptual distance calculator*; top-right in Figure 7.2). Moreover, the search space of FOL theories is intractable and hence, ILP resorts to employing "mode" definitions as search bias. However, we need stronger inductive biases in our setting. We allow for a stronger inductive bias by enabling the teacher to provide constraints as advice that can be directly added to the clausal theory. To find the most relevant constraints, GOCI queries the expert to choose among a sampled set of constraints (as first-order predicates) and uses the selected constraint to revise the

theory. This is analogous to a student asking relevant questions. Formally, we are investigating the following problem.

Given: A set of ground predicates (or trajectories) describing one (or few) instance(s) and expert guidance **To Do:** Induction of first-order logic representation of a concept that generalizes the given example(s) effectively.

Difference in Learning/Evaluation Paradigms: Our evaluation paradigm is distinct from that of ILP as well as traditional machine learning in general. Empirical evaluation, in traditional machine learning setting, happens by computing the predicted value of the target feature $\hat{y}_X$ for a test instance X represented as a feature vector of **fixed** length N. Similarly in case of ILP, we predict $\hat{y}_X$ for a test instance X, which here is represented as a **arbitrary** length vector. In GOCI, however, the notion of evaluation is quite different. A test instance X is represented as a planning task/goal and we evaluate it by testing if a valid plan π_X can be generated for the same (Figure 7.3). Consequently in learning, both traditional machine learning as well as ILP

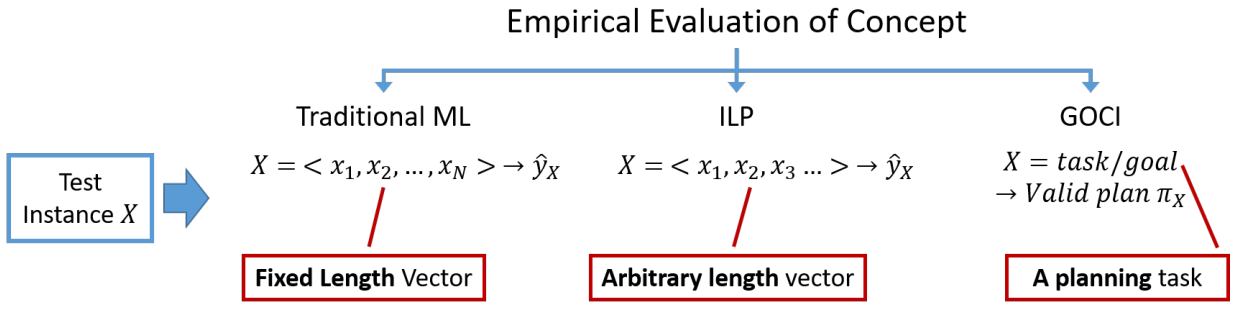


Figure 7.3: Differences between evaluation paradigm of our method GOCI and that of Traditional Machine Learning as well as ILP.

aim to maximize coverage of positive instances E^+ ($\max P(\hat{y}_x = true | y_x \in E^+)$) and minimize coverage of negatives E^-, (*i.e.* $\min P(\hat{y}_x = true | y_x \in E^-)$). GOCI evaluates a candidate concept representation by allowing the agent to realize that concept, for instance by computing a valid plan

for the goal/task implied by the instance x. Hence this is akin to plan induction, since we are learning parameterized plan for realizing the concept as a surrogate for the concept itself.

However, for the sake of theoretical analysis, we assume a traditional evaluation paradigm. Similar to ILP, we assume that the final/candidate theory representing a desired concept, learned by GOCI, is evaluated by predicting the outcome of a response variable $\hat{y}$, whose positive outcome implies that a given test instance X is indeed an instance of the desired concept. This allows us to derive algorithmic bounds on query budget, subject to a pre-defined error margin, under a PAC learning assumption.

7.3.1 Input

The input to GOCI is the **description of the instances(s)** of a concept that the human teacher provides. An input example of a concept is, thus, conjunction of a set of ground literals (assertions).

Example 13. *An instance of a concept in the domain of spatial structures, can be a structure* $\mathbb{L}$ *with dimensions* $height = 5, base = 4$ *(as shown in Figure 7.4).* $\mathbb{L}(S), Height(S, 5), Base(S, 4)$, s *is the concept identifier and may be described as conjunction of ground literals,*

$\texttt{Row(A)} \wedge \texttt{Tower(B)} \wedge \texttt{Width(A, 4)} \wedge \texttt{Height(S, 5)} \wedge \texttt{Base(S, 4)} \wedge \texttt{Contains(S, A)} \wedge$
$\texttt{Contains(S, B)} \wedge \texttt{Height(B, 4)} \wedge \texttt{SpRel(B, A,'NWTop');}$

which denotes $\mathbb{L}$ *as composition of a 'Row' of* $w = 4$ *and a 'Tower' of* $h = 4$ *with appropriate literals describing the scenario (Figure 7.4* ***left****). As a special case, under partial or total ordering assumptions among the ground literals, an input instance can represent a plan demonstration.*

7.3.2 Output

GOCI induces a least general generalization (LGG) horn clause from the *input* example(s).

Definition 12. *Concept in our setting is represented as disjunctive horn clause theory.*

$$T = \mathcal{C}(s^k \ldots) : - \bigvee_{k \in K} \left[\wedge_{i=1}^{N} f_i^k(t_1, \ldots, t_j) \right]$$

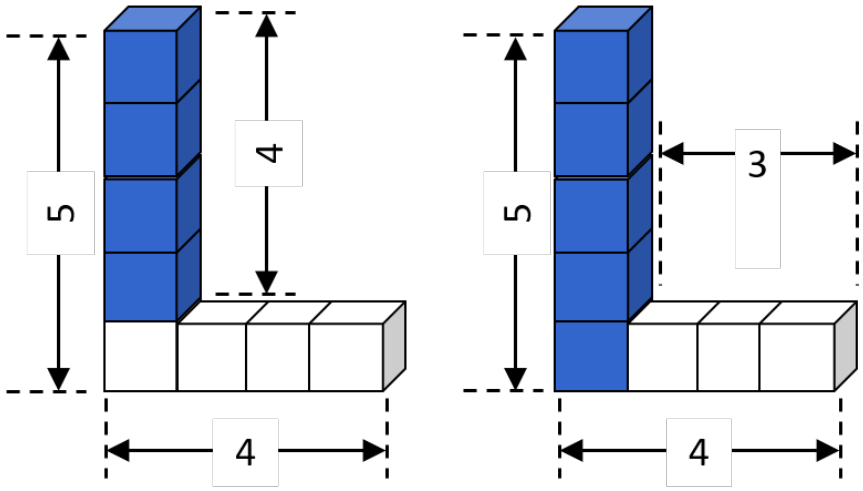


Figure 7.4: Concept $\mathbb{L}$ ($base = 4, height = 5$), described as composition of a **Tower** and a **Row**

where the body $\wedge_{i=1}^{N} f_i(t_1, \ldots, t_j)$ *is a conjunction of literals indicating known concepts, the head* $\mathcal{C}(s^k \ldots)$ *identifies a target concept, and the terms* $\{s^k\}$ *are logical variables that denote the parameters of the concept. Since a concept can be described in multiple ways (Figure 7.4), the final theory will be a disjunction over clauses with the same head. A (partial) instantiation of a theory* T *is denoted as* T/θ.

Note that these definitions allow for the reuse of concepts induced earlier, potentially in a hierarchical fashion. We believe that this is crucial in achieving human-agent collaboration in complex domains.

Example 14. *Figure 7.4 illustrates an instance of the concept* $\mathbb{L}$ *which can be described in multiple ways. A possible disjunction could be,*

$$\mathbb{L}(s) : -[\texttt{Height}(\texttt{s}, \texttt{h}_\texttt{s}), \texttt{Base}(\texttt{s}, \texttt{w}_\texttt{s}), \texttt{Contains}(\texttt{s}, \texttt{a}), \texttt{Contains}(\texttt{s}, \texttt{b}), \texttt{Row}(\texttt{a}), \texttt{Tower}(\texttt{b}),$$
$$\texttt{Width}(\texttt{a}, \texttt{w}_\texttt{a}), \texttt{Height}(\texttt{b}, \texttt{h}_\texttt{b}), \texttt{Equal}(\texttt{w}_\texttt{s}, \texttt{w}_\texttt{a}), \texttt{Sub}(\texttt{h}_\texttt{b}, \texttt{h}_\texttt{s}, 1), \texttt{SpRel}(\texttt{b}, \texttt{a}, \text{“}\texttt{NWTop}\text{”})]$$
$$\bigvee$$
$$[\texttt{Height}(\texttt{s}, \texttt{h}_\texttt{s}), \texttt{Base}(\texttt{s}, \texttt{w}_\texttt{s}), \texttt{Contains}(\texttt{s}, \texttt{a}), \texttt{Contains}(\texttt{s}, \texttt{b}), \texttt{Row}(\texttt{a}), \texttt{Tower}(\texttt{b}),$$
$$\texttt{Width}(\texttt{a}, \texttt{w}_\texttt{a}), \texttt{Height}(\texttt{b}, \texttt{h}_\texttt{b}), \texttt{Equal}(\texttt{h}_\texttt{s}, \texttt{h}_\texttt{b}), \texttt{Sub}(\texttt{w}_\texttt{a}, \texttt{w}_\texttt{s}, 1), \texttt{SpRel}(\texttt{b}, \texttt{s}, \text{“}\texttt{W}\text{”})]$$

The generalization must be clearly noted. The last argument of the `SpRel()` is a constant and not variable, since only this particular spatial alignment is appropriate for the concept of the $\mathbb{L}$ structure. Although the input is a single instance described in one way (Example 13), GOCI should learn a generalized representation such as Example 14. Another interesting aspect are the additional constraints: `Equal(X,Y)` and `Sub(X,Y,N)`. While such predicates are a part of the language, they are not typically described directly in the input examples. However, they are essential in appropriate generalization, since they can express complex interactions between numerical (non-numerical) parameters.

Note that we describe concepts in a generalized manner as horn clause theories. A specific instantiation could be plan induction from sparse demonstrations. In our approach, we can specify the time index as the last argument of both the state and action predicates. Following this definition, we can allow plan induction as shown in our experiments. Our novel conceptual distance is clearer and more intuitive in the case of plans as can be seen later.

Definition 13 (Decomposable:)**.** *A concept $\mathcal{C}$ is Decomposable iff it is expressed as a conjunction of other concepts, and one or more additional literals to model the interactions. $\mathcal{C} \Leftarrow (\bigwedge_i \mathcal{C'}_i) \wedge (\bigwedge_j B_j)$. Here $\mathcal{C'}_i$ are literals that represent other concepts and B_j are literals that, either describe the attributes of $\mathcal{C'}_i$ or connect them.*

Decomposable means that the concept (currently unknown) can be described as a composition of other known concepts. GOCI learns the class of decomposable concepts since it is intuitive for the "human teacher" to describe. Decomposable concepts faithfully capture the *modular* and *structured* aspect of how humans would understand and describe instances of concepts in the universe. It also connects to the decomposable nature of planning tasks which can be normally partitioned into sub-tasks that yield sub-plans.

7.3.3 Methodology

Search: Given the definition of the input and output spaces, we now describe the search process. ILP systems perform a greedy search through the space of possible theories. Space is typically defined, declaratively, by a set of mode definitions to guide the search process. Following most well-known ILP systems (Srinivasan, 2007), we also start with the most specific clause (known as a bottom clause) from the ground assertions and successively add/modify literals that might improve a rule that best explains the domain. Typically, the best theory is the one which most accurately explains positive example(s) provided while minimizing negative example coverage. Thus, it optimizes the likelihood of a theory T based on the data ($\mathbb{D} = E^+ \cup E^-$).

We start with a bottom clause, we variablize the ground statements via anti-substitution. Variabilization of theory T is denoted by $\theta^{-1} = \{a/x\}$ where $a \in consts(T)$, $x \notin vars(T)$. That is, inductive substitution θ^{-1}, is a mapping from occurrences of ground terms in theory T to variables.

Evaluation Score: To score a candidate theory, *it is necessary to modify the coverage-based ILP scoring* (e.g., ALEPH's compression heuristics) for the following reasons.

- To use as much of the user-provided advice as possible, we learn long theory, hence the *search space can be exponentially large*. Thus, modes alone aren't enough to guide our search strategy in this situation.

- There is only one (a few) positive training example(s) to learn from and *many possible rules can accurately match the training example*. Coverage based scores clearly fail here to identify the best theory.

Most inductive learners optimize some adaptation of likelihood. For a candidate theory T, likelihood given data $\mathbb{D}$ is, $LL(T) = \log P(\mathbb{D}|T)$, which is essentially its coverage. In our GOCI framework, we have one (at most few) positive example(s). Clearly coverage will not suffice. Hence, we define a modified objective as follows.

$$T^* = \underset{T \in \tau}{arg\,\min} \left(-LL(T) + D(T/\theta_X, X)\right) \tag{7.1}$$

where T^* is the optimal theory, τ is the set of all candidate theories, and D is the conceptual distance between the *instantiated* candidate theory T/θ_X and the original example X. As explained earlier, a theory $\mathbb{T}$ is a disjunction of horn clauses with the target concept as the head.

Distance metric: Conceptual distance, $D(T/\theta_X, X))$, is a penalty in our objective. The key idea is that any learned first-order horn clause theory must recover the given instance by equivalent substitution. However, syntactic measures, such as edit distance, are not sufficient since changing even a *single literal*, especially, literals that indicate inter-concept relations, could potentially result in a completely different concept. For instance, in blocks-world, the difference between a block being in the middle of a row and one at the end of the row can be encoded by changing one literal. Hence, a more sophisticated semantic distance such as conceptual distance is necessary (Friend et al., 2018). However, such distances are difficult to compute without a deeper understanding of the domain and its structure.

Our solution is to employ *inter-plan distances*. As discussed earlier, the class of concepts GOCI can induce, are decomposable and, hence, are equivalent to parameterized planning tasks. **One of the key contributions of this work is to exploit this equivalence by using a domain-independent planner to find grounded *plans* for both the theory learned at a particular iteration i, T_i and the instance given as input, X.** We then compute the Normalized Compression Distance (NCD) between the plans.

NCD for Plans: Goldman & Kuter (2015) proved that NCD is arguably the most robust inter-plan distance metric. NCD is a reasonable approximation of *Normalized Information Distance*, which by itself is not computable (Vitányi et al., 2009). Let the plans for T_i/θ_X and X be π_T and π_X. To obtain NCD, we execute string compression (lossy or lossless) on each of the plans as well as the concatenation of the two plans to recover the compressed strings C_T, C_X, and $C_{T,X}$ respectively. NCD between the plans can be computed as,

$$NCD(\pi_T, \pi_X) = \frac{C_{T,X} - \min(C_T, C_X)}{\max(C_T, C_X)} \tag{7.2}$$

The conceptual distance between a theory T and X is the NCD between the respective plans, $D(T/\theta_X, X) = NCD(\pi_T, \pi_X)$. This entire computation is performed by the *Conceptual distance calculator* as shown in Figure 7.2.

Observations about NCD: **(1)** Conceptual distance as a penalty term for negative log-likelihood in the scoring function ensures that the learned theory will correctly recover the given example/demonstration. **(2)** $D(T/\theta_X, X)$ generalizes to the *Kolmogorov-Smirnov* statistic between two target distributions if we induce probabilistic logic theories. We prove these insights theoretically in the next section.

Human Guidance: As outlined earlier, the search space in ILP is provably infinite. Typically language-bias (modes) and model assumptions (closed world) are used to prune the search space. However, it is still intractable with one (or few) examples. Hence, we employ human expert guidance as constraints that can be directly used to refine an induced theory, acting as a strong inductive bias. Also, we are learning decomposable concepts (see Definition 13). This allows us to exploit another interesting property. Constraints can now be applied over the attributes of the known concepts that compose the target concept, or over the relations between them. Thus, GOCI directly includes such constraints in the clauses as literals (see Example 14). Though such constraint literals come from the pre-declared language, they are not directly observed in the input example(s). So an ILP learner will fail to include such literals in the induced clauses.

If the human inputs (constraints) are provided upfront before learning commences, it turns out to be wasteful/irrelevant. More importantly, it places the burden on the human consider all possible scenarios where advice could be needed. To alleviate this, *our framework explicitly queries for human advice on the relevant constraint literals, which are most useful*. Let $\mathbb{U}$ be a predefined library of constraint predicates in the language, and let $\mathcal{U}() \in \mathbb{U}$ be a relevant constraint literal. Human advice $\mathcal{A}$ can essentially be viewed as a preference over the set of relevant constraints $\{\mathcal{U}()\}$. If $U_{\mathcal{A}}$ denotes the preferred set of constraints, then we denote the theory having a preferred constraint

literal in the body of a clause as $\tau_{\mathcal{A}}$. *(For instance, based on Example 14* GOCI *could query which of the two sampled constraints* $\texttt{Sub}(\texttt{h}_\texttt{b}, \texttt{h}_\texttt{s}, 1)$ *&* $\texttt{Greater}(\texttt{h}_\texttt{b}, \texttt{h}_\texttt{s})$ *is more useful. Human could prefer* $\texttt{Sub}(\texttt{h}_\texttt{b}, \texttt{h}_\texttt{s}, 1)$*, since it clearly subsumes the other.)* Scoring function now becomes:

$$T^* = \underset{T \in \tau}{arg\,\min} \left(-LL(T) + D(T/\theta_X, X)\right) \ : \tau \subseteq \{\tau_{\mathcal{A}}\}$$

Thus, we are optimizing the constrained form of the same objective as Equation 7.1 which aims to prune the search space. This is inspired by advice elicitation approaches (Odom et al., 2015). While our framework can incorporate different forms of advice, we focus on preference over constraints on the logical variables, in our one-shot setting. This relates to the work on learning with qualitative constraints by Yang et al.,(2013). The formal algorithm, described next, illustrates how we achieve this via an iterative greedy refinement (Figure 7.2, query-advice loop shown in left part of the image).

7.3.4 The GOCI Algorithm

Algorithm 10 outlines the GOCI framework. It initializes a theory T_0, by variablizing the *'bottom clause'* obtained from X and background knowledge [`lines 3 & 5`]. Then it performs a standard ILP search (described earlier) to propose a candidate theory [`line 6`]. This is followed by the guided refinement steps, where constraint literals are sampled (parameter tying guides the sampling) and the *human teacher* is queried for preference over them, such that the candidate theory can be modified using preferred constraints [`lines 7-9`]. The function NCD() performs the computation of the conceptual distance, by first grounding the current modified candidate theory T' with the same parameter values as the input example X, then generating grounded plans and finally calculating the normalized compression distance between the plan strings (as shown in Figure 7.2 and Equation 7.2) [`line 10`]. Distance-penalized negative log-likelihood score is estimated and minimized to find the best theory at the current iteration [`lines 11-14`], which is then used as the initial model in the next iteration. This process is repeated either until convergence (no change in induced theory) or maximum iteration bound (L).

Algorithm 10 Guided One-shot Concept Induction

procedure GOCI(Instance X)
  Initialize: Set Iteration $\ell \leftarrow 1$
  Initialize: Bootstrap theory $T_0 \leftarrow X/\theta^{-1}$
  repeat
    Use $T_{\ell-1}$ as initial model
    Candidate theory $T_\ell \leftarrow \text{SEARCH}(T \in \tau | T_{\ell-1})$
    Sample applicable constraints $\mathcal{U} \in \mathbb{U}$
    $\mathcal{U}_\mathcal{A} \leftarrow \text{QUERY}(human, \mathcal{U})$
    $T' \leftarrow T_\ell \oplus \mathcal{U}_\mathcal{A}$ ▷ $\forall\, \mathcal{U}_\mathcal{A} \in \mathcal{A}$
    $D_\ell(T'/\theta_X, X) \leftarrow \text{NCD}(\pi_{T'/\theta_X}, \pi_X)$
    Score $\mathcal{S}_\ell \leftarrow (-LL(T') + D(T'/\theta_X, X))$
    if $S_\ell < S_{\ell-1}$ **then** ▷ minimize
      Retain T': Update $T_\ell = T'$
    end if
  until $\ell \leq L$ OR $T_\ell = T_{\ell-1}$
end procedure

7.3.5 Theoretical analysis

Validity of Distance Metric:

Normalized information distance $\delta(x, y)$ between 2 strings x and y is provably a valid distance metric (Vitányi et al., 2009) (under reasonable assumptions),

$$\delta(x, y) = \frac{\max K(x|y), K(y|x)}{\max K(x), K(y)}$$

where $K(x)$ is the Kolmogorov complexity of a string x and $K(x|y)$ is the conditional Kolmogorov complexity of x given another string y. NCD is a computable approximation of the same [$D(x, y) \approx \delta(x, y)$]. Thus, we just verify if δ is a correct conceptual distance measure.

Consider two theories T_Y and T_Z, with same parameterizations (*i.e.*, same heads). Let T_Y/θ and T_Z/θ be their grounded theories with identical parameter values θ. Our learned theories are equivalent to planning tasks. Assuming access to a planner $\Pi()$ which returns $Y = \Pi(T_Y/\theta)$ and $Z = \Pi(T_Z/\theta)$, the two plan strings w.r.t the instantiations of concepts represented by T_Y and T_Z respectively.

Proposition 3 (Valid Conceptual Distance)**.** *Normalized information distance $\delta(Y, Z)$ is a valid and sound conceptual distance measure between T_Y and T_Z, i.e., $\delta(Y, Z) = 0$ iff the concepts represented by T_Y and T_Z are equivalent.*

$$(\delta(Y, Z) = 0) \iff (T_Y \equiv T_Z)$$

Proof Sketch for Proposition 3: Let T_Y and T_Z be 2 induced consistent first-order Horn clause theories, that may or may not represent the same concept. Let θ be some substitution. Now let T_Y/θ and T_Z/θ be the grounded theories under the same substitution. This is valid since we are learning horn clause theories with the same head, that indicates the target concept being learned. As explained in the the manuscript a theory is equivalent to a planning task. We assume access to a planner $\Pi()$, we get plan strings $Y = \Pi(T_Y/\theta)$ and $Z = \Pi(T_Z/\theta)$ with respect to the planing tasks T_Y/θ and T_Z/θ.

Friend et al.,(2018) proved that *Conceptual Distance* is the step distance between 2 consistent theories in a cluster network $(\mathbb{T}, \rightleftarrows, \sim)$, where $\mathbb{T}$ is the class of consistent theories, $\rightleftarrows$ is the definitional equivalence relation (equivalence over bidirectional **concept extensions**) and $\sim$ implies symmetry relation. We have shown in the paper that, given the class of concepts we focus on, *a concept is a planning task.*

Let there be a theory T^*, which represents the optimal generalization of a concept $\mathcal{C}$. If step distance $\langle T_Y, T^* \rangle = 0$ in a cluster network and $\langle T_Z, T^* \rangle = 0$ then $\langle T_Y, T_Z \rangle = 0$, *i.e.* they represent the same concept C and they are definitionally equivalent $T_Y \rightleftarrows T_Z$. Thus both T_Y/θ and T_Z/θ will generate the same set of plans as T^*, since they will denote the same planning tasks (By structural induction). Thus,

$$T_Y \rightleftarrows T_Z \iff [\Pi(Y) \cap \Pi(Z) = \Pi(Y) = \Pi(Z)] \tag{7.3}$$

upto equivalence of partial ordering in planning. let $\pi^*()$ be a minimum length plan in a set of plans $\Pi()$. Let y and z be strings indicating plans $\pi^*(Y)$ and $\pi^*(z)$ ignoring partial order. If $\Pi(Y) = \Pi(Z)$ then $\pi^*(Y) = \pi^*(z)$.

So, conditional Kolmogorov complexities $K(y|z) = 0$ and $K(z|y) = 0$, if the strings are equivalent ignoring partial ordering since a universal prefix truing machine will recover one string given the other in 0 steps.

$$\therefore \frac{max\,(K(y|z), K(z|y))}{max\,(K(y), K(z))} = 0 = \delta(Y, Z)$$

Proposition 4 (Generalization to Kolmogorov-Smirnov statistic)**.** *In generalized probabilistic logic, following Vitányi (2013), $\delta(Y, Z)$ corresponds to two-sample Kolmogorov-Smirnov statistic between two random variables T_Y/θ and T_Z/θ with distributions P_{T_Y} and P_{T_Z} respectively. $v(T_Y, T_Z) =$ **sup**$_{\theta\in\mathcal{F}} |F_{T_Y}(\theta) - F_{T_Z}(\theta)|$, where $F_{T_Y}()$ is the cumulative distribution function for P_{T_Y} and **sup**$_{\theta\in\mathcal{F}}$ is the supremum operator. In deterministic setting, δ is a special case of v, $\delta(Y, Z) \preceq v(F_{T_Y}, F_{T_Z})$.*

Proof Sketch for Proposition 4: This can be proved by considering the connection between NID and the distributions induced by the concept classes we are learning. NID is defined as $\delta(x, y) = \frac{\max K(x|y), K(y|x)}{\max K(x), K(y)}$, where, $K(a|b)$ is the conditional Kolmogorov complexity of a string a, given b. There is no provable equivalence between Kolmogorov complexity and traditional notions of probability distributions.

However, if we consider a ***reference*** universal semi-computable semi-probability mass function $\mathbf{m}(x)$, then there is a provable equivalence $-\log \mathbf{m}(x) = K(x) \pm O(1)$. Similarly for conditional Kolmogorov complexity, by Conditional Coding Theorem, $-\log \mathbf{m}(y|x) = K(y|x) \pm O(1)$ (Vitányi, 2013). By definition,

$$m(y|x) = \sum_{j\geq 1} 2^{-K(j)-c_j} P_j(y|x)$$

where $c_j > 0$ are constants and $P_j(y|x)$ is the lower semi-computable conditional. A lower semi-computable semi-probability conditional mass function is based on the string generating complexity of a **universal prefix Turing machine**. Thus $m(y|x)$ is greater than all the lower semi-computable. Note that our compressed plans are equivalent to a string generated by Universal

Turing Machines. The conditional case implies, if a compressed plan string x is given as an auxiliary **prefix** tape, how complex it is to generate compressed string $y = \theta$.

Given 2 grounded theories T_Y/θ and T_Z/θ, let $P_{T_Y/\theta}$, $P_{T_X/\theta}$ be the respective distributions when learning probabilistic logic rules. Now let us define the semantics of a distribution $P_{T/\theta}$ in our case: $P_{T/\theta} = P(\pi(T/\theta))$, *i.e.* distribution over the plan strings, which can be considered as lower semi-computable probability based on coding theory. We know,

$$\sum_{j \geq 1} 2^{-K(j)-c_j} P_j(y) \approx F(y|x) \tag{7.4}$$

where $F(y)$ is the cumulative distribution So, NID $\delta(Y, Z)$ now becomes,

$$\delta(Y, Z) = \frac{max\left(K(y|z), K(z|y)\right)}{max\left(K(y), K(z)\right)}$$

We know that $max\left(K(y), K(z)\right)$ is a normalizer. Thus, $\delta(Y, Z) < max\left(K(y|z), K(z|y)\right)$

$$max\left(K(y|z), K(z|y)\right) = max\left(-logm(y|z), -logm(z|y)\right)$$
$$= max\left(-log\frac{m(y,z)}{m(z)}, -log\frac{m(y,z)}{m(y)}\right)$$
$$= max\left(\left[-\log m(y,z) + \log m(z)\right], \left[-\log m(y,z) + \log m(y)\right]\right)$$

Under partial ordering max yields supremum

$$\approx \mathbf{sup}\left|\log m(y) - \log m(z)\right|$$
$$\approx \mathbf{sup}\left|\log F(y) - \log F(z)\right|$$
$$\approx \mathbf{sup}\left|F(y) - F(z)\right| \quad \text{[log is monotonic]}$$

PAC Learnability:

To analyze the theoretical properties of GOCI, we build on the PAC analysis for recursive *rlgg* (relative least general generalization) in GOLEM for function-free *horn* clause induction due to

Muggleton & Feng (1990). Let n denote the sample size and $\mathcal{H}$ denote the hypothesis space. As shown earlier, GOCI learns the final theory by iterative refinement after inducing an initial theory. Denote the initial hypothesis space as $\mathcal{H}_0$ and hypothesis space of the final theory as $\mathcal{H}^*$ (such that $T^* \in \mathcal{H}^*$).

Proposition 5 (Sample complexity)**.** *Following Valiant (1984) and Mooney (1994), with probability* $(1-\delta)$*, the sample complexity of inducing the optimal theory* T^* *is:*

$$n^* = \mathcal{O}\left(\frac{1}{\epsilon}\left[d^L \ln(|\mathcal{H}_0| + d + m) + \ln(\frac{1}{\delta})\right]\right) \tag{7.5}$$

where ϵ *is the regret,* n^* *- sample complexity of* $\mathcal{H}^*$*,* m *- number of distinct predicates,* d *is the distance of the current revision from the last known consistent theory, and* L *is the upper bound on the number of refinement steps (iterations).*

ILP induces ij-Determinate clauses (Muggleton and Feng, 1990), where i is the maximum depth of the clause & j is the maximum arity. Thus, in our problem setting, $|\mathcal{H}_0| = \mathcal{O}((tpm)^{j^i})$, where t is the number of terms, and p is the place (Muggleton and Feng, 1990). Hence, (if j & i is bounded : $j^i = c$) Equation 7.5 can be reformulated as:

$$n^* = \mathcal{O}\left(\frac{1}{\epsilon}\left[d^L \ln\left(((tfm)^c) + d + m\right) + \ln(\frac{1}{\delta})\right]\right) \tag{7.6}$$

Mooney (1994) defines distance d to be the number of single literal changes in a single refinement step. In Algorithm 10, we observe that at each iteration $\ell \leq L$, updates are w.r.t. the preferred constraint predicates $\mathcal{U}_\mathcal{A} \in \mathbb{U}$.

Proposition 6 (Refinement distance)**.** d *is upper bounded by the expected number of literals that can be constructed out of the library of constraint predicates with human advice* $\mathbb{E}_{\sim\mathcal{A}}\left[|\mathbb{U}|\right]$ *and lower bounded by the conceptual distance between theory learned at two consecutive iterations since we adopt a greedy approach. If* $Pr_\mathcal{A}(\mathcal{U})$ *denotes the probability of a constraint predicate being preferred then,* $|D_\ell - D_{\ell-1}| \leq d \leq \sum_{i=1}^{2^{(|\mathbb{U}|-1)} \times {}^tP_q} Pr_\mathcal{A}(\mathcal{U}_i)$ *where* $2^{(|\mathbb{U}|-1)} \times {}^tP_q$ *is the maximum possible number of constraint literals and* q *is the maximum arity of the constraints. If we use only pairwise constraints, then* q =2*.*

Our input is sparse (one or few instances). GOCI elicits advice over constraints to acquire additional information. Let $|X|$ be the number of input examples.

Proposition 7 (Advice complexity)**.** *From Equations 7.5 and 7.6, at convergence $\ell = L$, we get $\frac{n^* - |X|}{L}$ examples, on an average, for a concept $\mathcal{C}$ to be PAC learnable using* GOCI.

7.4 Experimental Evaluation

We next aim to answer the following questions explicitly:

(Q1) Is GOCI effective in one-shot concept induction?

(Q2) How sample efficient is GOCI compared to baselines?

(Q3) What is the relative contribution of the novel scoring function vs. human guidance towards performance?

We developed our framework by extending Wisconsin Inductive Logic Learner (Natarajan et al., 2009). We modified the scoring function with NCD penalty and integrated a customized SHOP2 planner (Nau et al., 2003) for inter-plan NCD computation. We added constraint sampling and human guidance in iterative fashion as outlined in Algorithm 10.

7.4.1 Experimental Setup

We compare GOCI with a standard ILP system with no enhancements. We focus on the specific task of "one-shot concept induction", with a single input example for each of the several types of concepts and report aggregated precision. We consider precision because preference queries are meant to eliminate false positives in our case. To demonstrate the robustness of GOCI w.r.t sample complexity, beyond one-shot case, we conducted experiments with varying sample sizes for each concept type across all domains and show learning curves for the same. We also evaluate the relative contribution of two important components of our proposed framework: (a) novel scoring

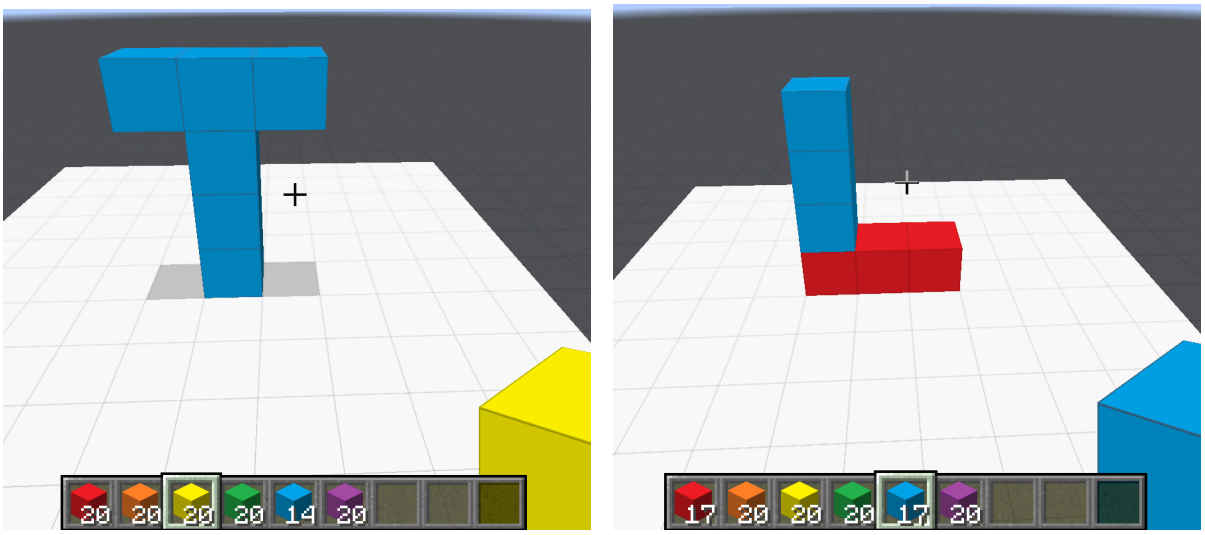


Figure 7.5: Instances of spatial concepts in Minecraft. *(Left)* Upright Tee, *(Right)* Upright L

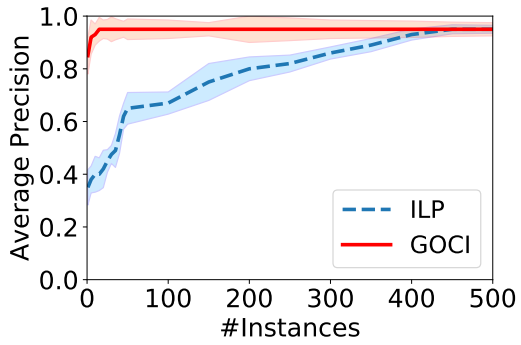
metric and (b) human guidance; *i.e.*, we compare against two more baselines (*ILP+Score* and *ILP+Guidance*). For every domain, we consider ten different types of concepts (ten targets). All the results are aggregated over five different runs.

Domains We employ 3 domains with varying complexity:

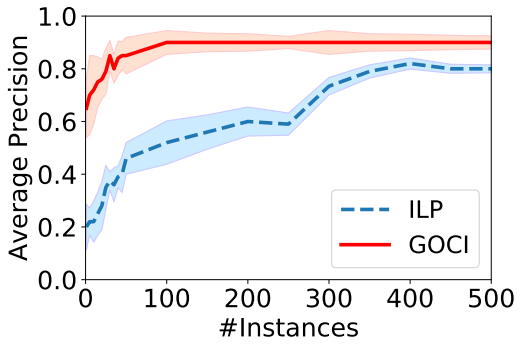
1. Minecraft (Spatial Structures): The goal is to learn discrete spatial concepts in a customized (Narayan-Chen et al., 2019) Project Malmo platform for Minecraft. Dialogue data in Malmo is available online, and we converted them into a logical representation. All structures are in terms of discrete atomic unit blocks (cubes). Figure 7.5 shows examples of some simple spatial structures that GOCI was able to learn. The representation language is similar to the Example 14, with some pre-existing concepts in knowledge base such as `Row`, `Tower`, `Cube` etc.

2. Assembly (a planning domain): Generalized concepts are equivalent to parameterized planning tasks. Assembly is a well-known planning domain, where different mechanical structures (machines) are constructed using different parts and resources. Input is a conjunction of ground literals indicating ground plan demonstration (assuming total ordering).

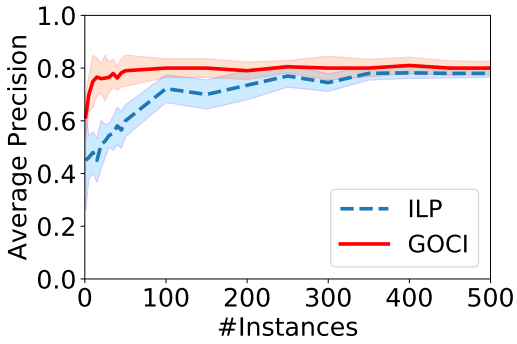
3. Chemical Entities of Biological Interest (ChEBI): ChEBI is a compound database which contains some important structural features and activity-based information for classification of



(a) Minecraft domain



(b) Assembly domain



(c) ChEBI

Figure 7.6: Learning curves for varying sample size comparing sample-efficiency of GOCI and ILP (best viewed in color).

chemicals. Core information ChEBI provides are (1) Molecular structure, (2) Biological role, (3) Application, and (4) Subatomic particle, which may be suitable for the prediction of various attributes of compounds. We model the Benzene molecule prediction task following the description in ChEBI. It has predicates such as `Carbon`, `SingleBond`, `DoubleBond`, `HasAtom` etc.

7.4.2 Experimental Results

[Effective One-shot (Q1)] Table 7.1 shows the performance of GOCI on one-shot concept learning tasks as compared to standard ILP. GOCI significantly outperforms ILP across all domains answer-

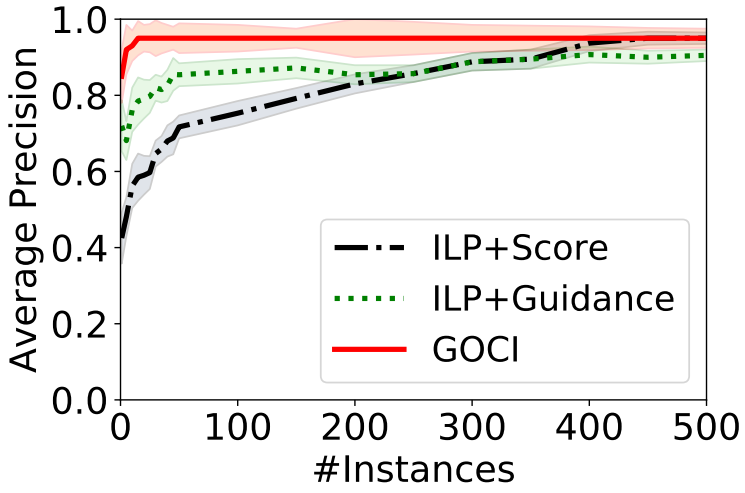


Figure 7.7: Results of ablation study on Minecraft domain. Relative contribution of our distance-penalized score vs. human guidance (best viewed in color).

Table 7.1: Results for one-shot concept learning.

Domain	Approach	Avg. Precision	#Queries
Minecraft	GOCI	0.85	5.5 ± 3
	ILP	0.35	-
Assembly	GOCI	0.65	16.5 ± 4
	ILP	0.2	-
ChEBI	GOCI	0.615	13.1 ± 2.13
	ILP	0.45	-

ing (Q1) affirmatively. Also, note that GOCI is very 'query' efficient as observed from the small average number of queries posed in the case of each domain.

[Sample Efficiency (Q2)] In figure 7.6, we observe that GOCI converges within significantly smaller sample size across all domains, thus, supporting our theoretical claims in Section 7.3.5. In ChEBI, though, quality of planning domain encoding might explain mildly lower-precision yet GOCI does perform significantly better than vanilla ILP learner.

[Relative contribution (Q3)] Figure 7.7 validates our intuition that both components (scoring function and human-guidance) together make GOCI a robust one-shot (sample-efficient) concept induction framework. Though human guidance, alone, is able to enhance the performance of a

vanilla ILP learner in sparse samples, yet it is not sufficient for optimal performance. In contrast, although the advantage of our novel distance-penalized scoring metric is marginal in sparse samples, it is essential for optimal performance at convergence.

The most important conclusion from the experiments is that when available, the guidance along with the score leads to a **jump-start, better slope and in some cases, asymptotically sample efficient with a fraction of the number of instances needed** than merely learning from data. This clearly demonstrates the need for the injection of human guidance as knowledge when learning complex concepts.

7.5 Conclusions

We developed a human-in-the-loop one-shot concept learning framework in which the agent learns a generalized representation of a concept as FOL rules, from a single (or a few) positive example(s). We make two specific contributions – deriving a new distance measure between concepts and allowing for richer human inputs than mere labels to be solicited actively by the agent. Our theoretical and experimental analyses showed the promise of GOCI method.

CHAPTER 8

ADDITIONAL EXPLORATION

This chapter presents some additional research work in which I made significant contributions. All of these are closely connected to the primary objectives of this dissertation. Setion 8.1 presents GMC (Hayes et al., 2017), an intuitive interactive visual interface for automatic elicitation of background knowledge from domain experts. Such an interface could potentially prove to be a crucial component of a true HAAI framework. Section 8.2 describes the SKID^3 (Dhami et al., 2018) a framework that addresses the important real-world problem of effectively identifying and discovering adverse effects due to interaction of multiple drugs administered simultaneously. This is critical in a treatment planning scenario (Example 1 in Chapter 1). Our novel approximation techniques have also been utilized for efficient grounding in this work. Section 8.3 outlines DACBOOST (Ramanan et al., 2019) that proposes a gradient-boosted non-parametric learning approach for learning tractable probabilistic models, such as Arithmetic Circuits. Tractable modeling is crucial for scaling in data-dense environments. Finally, Section 8.4 briefly outlines all the other frameworks that adopt our count approximation approaches (FACT and MACH) for efficiency.

8.1 Automatic Construction of Background Knowledge

As we have discussed in the earlier chapters, ILP systems (Srinivasan, 2004) can effectively induce generalized compact patterns from data in the form of FOL clauses. They search through the space of candidate clauses, scoring and modifying them in a greedy fashion. For instance, Aleph (Srinivasan, 2004) generates clauses bottom-up by constructing the most specific explanation of examples and then generalizing while TILDE (Blockeel, 1999) constructs clauses top-down. However, the search space in ILP is extremely large and essentially intractable. This problem is alleviated by use of background knowledge, known as *'modes'*, based on the given domain. Modes can be considered as additional directives that can prune, constrain or characterize the search space mak-

ing efficient search possible. But the effective design of modes is not trivial and requires a domain expert to have ILP expertise as well.

Guided Mode Construction (GMC) by Hayes et al., (2017), relaxes the assumption of ILP expertise, and develops a user-friendly framework for a domain expert to provide relevant domain knowledge which is then automatically transformed into accurate mode definitions. One of the key contribution of this framework is the intuitive graphical user interface that allows a domain expert to visualize the domain structure via Entity-Relationship Diagrams (ERD) and use visual annotations to convey knowledge. It also proposes a novel approach to transform such visual annotations into mode definitions via path-finding on the annotated ERDs.

This work is very important in the context of SRL as well since search strategies in SRL are inspired from ILP (Natarajan et al., 2015, 2010). *Modes*, like in ILP, restrict the search space making the learning of probabilistic clauses efficient. It has been observed that many real users of these systems, especially those who fail to learn good models with these algorithms, may not able to select the correct modes to guide the search and are, thus, unable to effectively use such models in real tasks. Integrating databases underneath the learner to improve the search speed (Malec et al., 2016; Zeng et al., 2014; Niu et al., 2011) is one of the ways that people have addressed this challenge. While these systems have certainly improved the search, recent work by Malec et al. (Malec et al., 2016) clearly demonstrated the need for modes to achieve effective learning even when using databases. Hayes et al., (2017) instead automatically converts the user annotated ER diagrams to mode definitions that are then later employed to guide the search. It closely relates to the work of Walker et al. (2011) where a UI was designed to provide *advice* for a PLM learner. Their UI was domain-specific, whereas the approach showed in Hayes et al., 2017 is a generalized one to utilize ER diagrams to automatically construct modes for logic-based learners.

The Entity-Relational Model: The entity-relationship model (Chen, 1976) allows for intuitively expressing and visualizing the structure and semantics of a database at an abstract level

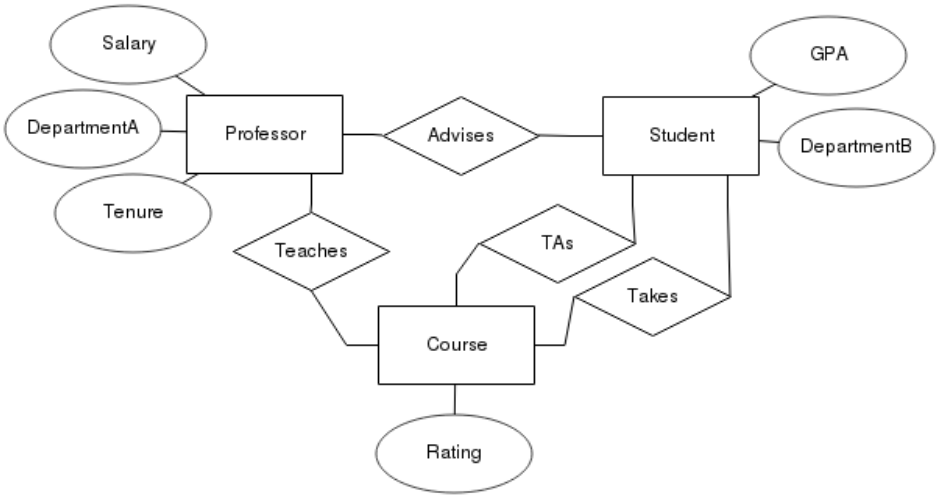


Figure 8.1: An ER-Diagram illustrating 3 entities, *Professors, Students, and Courses*, their attributes (ovals) and the relationships (diamonds) among them.

as objects and classes of objects (entities and entity classes), attributes of such entities, and relationships that exist between such entity classes in the form of a graphical diagram (Garcia-Molina et al., 2009). Figure 8.1 illustrates an ER diagram for an example *university* domain involving professors, students and courses and the relationships they share.

One key intuition behind this work is the connection between constrained logical clause search and SQL (Structured Query Language) query augmentation. For instance, "Hints," in SQL queries, are special symbolic tools to guide the query evaluation engine to prioritize some database operation over the others to enhance efficiency (Lohman et al., 2000; Diab et al., 2009; Bruno et al., 2012). Thus, they are akin to soft directives/constraints (modes) on the search space.

Modes in ILP: Modes in ILP allows us to describe the underlying domain structure as well as constrain the search space and is hence a key aspect of effective learning in SRL. A mode for predicate $pred$ with n arguments is defined as,

$$\texttt{pred}\,([+/-/\#]\texttt{type}_1, [+/-/\#]\texttt{type}_2, ..., [+/-/\#]\texttt{type}_\texttt{n})$$

. Each type indicates the entity class which can appear as that argument. The preceding symbols indicate if it can be tied with an input variable (+), an output variable (-), or a constant (#). Input

variables must be previously defined in the model. Output variables are free variables that have not been defined, essentially serving the purpose of constraining the search space. Hayes at al., 2017 automatically constructs such modes and can be adapted to any ILP/PLM systems.

8.1.1 Automatic Human-Guided Mode Construction

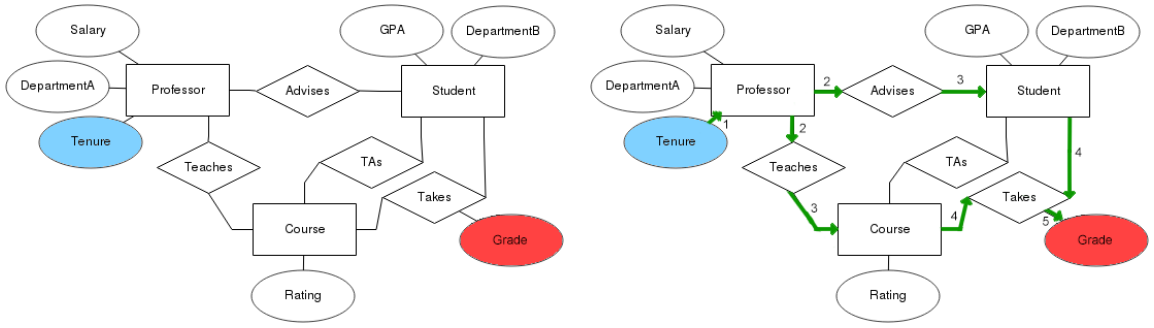
In the (GMC framework, the human is assumed to be a domain expert and not an ILP expert. It has the following key steps:

1. The GUI module provides the expert an interactive visualization of the domain structure with an ER diagram. The expert can select and annotate any component of this ERD, including the target entity/relation (about which the model is to be learned), as well as the entities/attributes that are supposed to be relevant to modeling the target.

2. Then it discovers directed paths on the annotated ERD between the target entity/relations and the other entities/attributes that has been marked relevant by the user.

3. Te discovered paths are used to construct modes. The edges in paths due to the Relations and attributes are encoded as predicates, the involved entity classes become the arguments of those predicates, thus defining the types. The indicators for variable tying/sharing is obtained from the ordering in the directed paths. The arguments in the target predicate are always set as closed/input (+). Following that, as we move along a path starting from when an entity class is encountered for the first time it is set to open/output (-) encouraging the learner to use assign new variables during search. Any subsequent occurrence of the same entity class along the path leads to it being assigned a closed (+) indicator, forcing a previous variable to be used during search. An attribute it will be assigned the # mode.

Figure 8.2(a) shows such an annotated ERD (Step 1), for our example university domain, where *Grade* (marked in red) was identified by an expert as being an important attribute for predicting



(a) *Target*: 'Tenure'. *Informative/important*: 'Grade', selected by user.

(b) The equidistant shortest paths between Tenure and Grade.

Figure 8.2: Illustrative example showing knowledge guided walks on the ERD, given in Figure 8.1, for mode construction.

Table 8.1: Each step of a path and corresponding modes generated by GMC.

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
Path 1	$Tenure$	$Professor$	$Advises$	$Student$	$Takes$	$Grade$
Modes 1	$Tenure(+Prof)$		$Advises(+Prof, -Stud)$		$Takes(+Stud, -Course, \#Grade)$	
Clause 1	$Tenure(p) \wedge Advises(p, s) \wedge Takes(s, c, A+)$					

Tenure (marked in blue). GMC first connects the target concept to the related concepts by finding paths from one to another in the ER diagram. Figure 8.2(b) shows two paths that connect *Tenure* to *Grade*.

Consider the example domain involving professors, students and courses, shown earlier. Figure 8.2(a) shows an annotated ERD where *Grade* (marked in red) was identified by an expert as being an important attribute for predicting the target attribute *Tenure* (marked in blue). GMC first connects the target concept to the related concepts by finding paths from one to another in the ER diagram. Figure 8.2(b) shows two paths that connect *Tenure* to *Grade* (Step 2).

Table 8.1 shows each step in one of the paths and how the corresponding modes are constructed from the same (Step 3). Entities and attributes are highlighted in different colors to show which arguments have the same type. The first time a type ($Student, Course$) is introduced along the path, the mode is set to $-$ allowing a free variable to be introduced and all subsequent appearances of the same are set to $+$. Clause 1 in Table 8.1 gives an example of a clause that could be generated by an ILP system with the specified modes. The English interpretation of this rule is that tenure

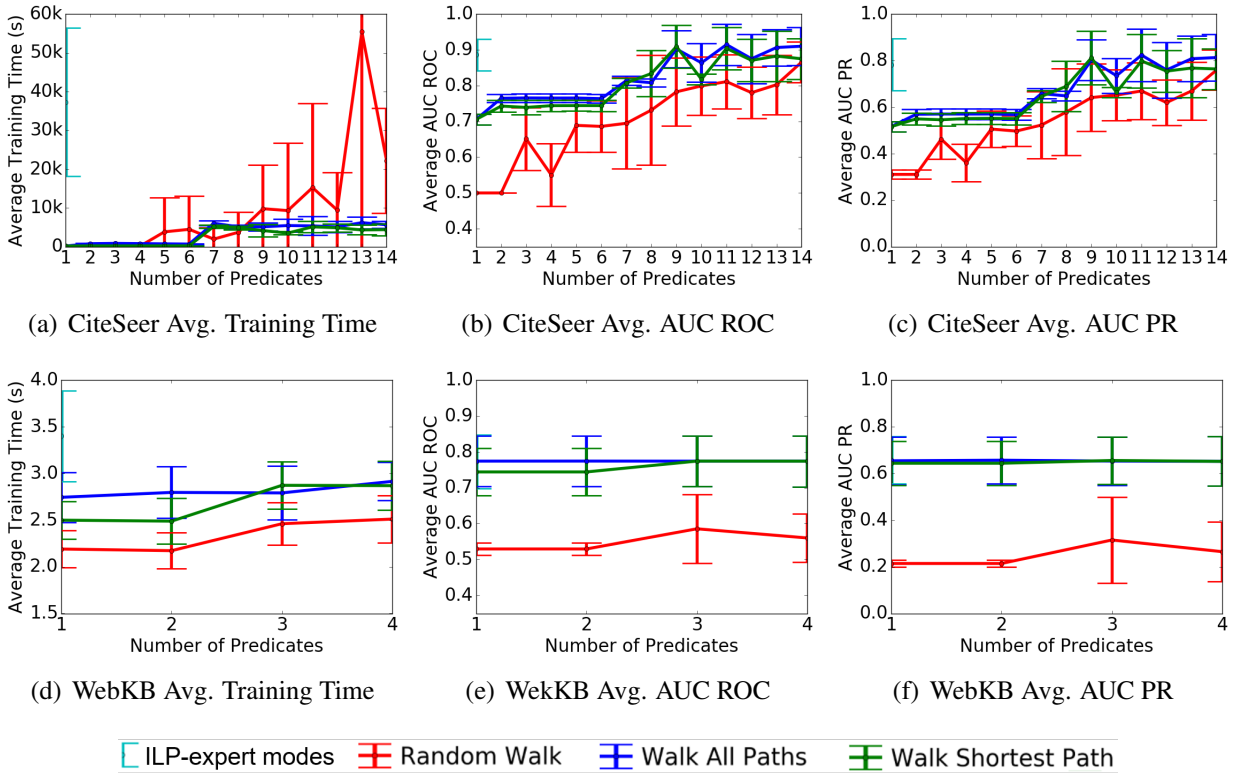
depends on the grades of students who are advised by a professor (For further details about the GMC algorithm please refer to Hayes et al.,2017)

8.1.2 Experimental Evaluation

Hayes et al., (2017) shows, through substantial empirical evaluations, how (GMC) can effectively generate modes that are comparable to or better than the optimal modes provided by an ILP expert or modes generated by random paths without expert annotation. We have used the state-of-the-art ILP structure/parameter learning framework Relational Functional Gradient Boosting (Natarajan et al., 2010) as the test-bed for evaluating the quality of automatically constructed modes.
Domains: While, Hayes et al., (2017) use four standard ILP/PLM datasets, namely CiteSeer (Poon and Domingos, 2007), WebKB (Mihalkova and Mooney, 2007), Cora (Poon and Domingos, 2007), and IMDB (Mihalkova and Mooney, 2007) for empirical evaluation, I present the results on 2 of them (Citeseer and WebKB) in this chapter. GMC framework itself has two different settings, `Walk All Paths` for walking all paths on the graph from the user-specified target to each selected feature, and `Walk Shortest Path` for finding only a shortest path from the target to each selected feature. In the experimental results (Figures 8.3), the **x-axis** represents the number of concepts (entity classes/relations/attributes) marked as relevant/important towards modeling the target concept, in the order that they were marked by the user. This shows how inclusion each additional relevant predicate influences performance/training time.

Figure 8.3 presents the experimental results (on two of the domains). We can make the following observations from the training time and performance plots:

1. **Effective:** Both settings of GMC outperforms the baseline modes set by *Random Walk*. The difference is more significant earlier in the learning curve when fewer paths are being found. Additionally, both settings of GMC exhibit comparable performance against *ILP-expert defined modes*. This, GMC generates modes that leads to effective search.



(a) CiteSeer Avg. Training Time (b) CiteSeer Avg. AUC ROC (c) CiteSeer Avg. AUC PR

(d) WebKB Avg. Training Time (e) WekKB Avg. AUC ROC (f) WebKB Avg. AUC PR

Figure 8.3: Results for CiteSeer and WebKB Datasets; Top row: Citeseer, Bottom row: WebKB. Left: Efficiency - Training time (lower is better), Middle & Right: Performance - Average AUC ROC and AUC PR respectively (higher is better).

2. **Efficiency:** GMC, can significantly constrain the search space which allows for efficient modeling. Figures 8.3(a) & 8.3(d) show the average training time in the two domains. Clearly, GMC exhibits significantly lower training time than ILP-expert modes as well as *Rendom Walks* in *Citeseer*, especially with higher number of relevant predicates. In WebKB, although *Random Walks* take less time, we should note that the respective performance is significantly worse, showing the *Random Walks* are not generating optimal modes.

3. **Significance of Human Guidance:** The empirical results illustrate that, there exists a convergence point where including additional guidance (annotations of important predicates) no longer leads to better performance. The domain expert is essential for identifying and annotating what the most important features/nodes are.

Thus GMC represents one significant step towards developing a visual intuitive interface that will allow of seamless bidirectional communication between human(s) and the AI agent(s) in our envisioned HAAI framework.

8.2 Drug-Drug Interaction prediction and discovery

Drug-drug interactions (DDIs) occur when multiple medications are co-administered and can potentially cause adverse effects on the patients. DDIs have emerged, around the world, as a major cause of hospital admissions, rehospitalizations, emergency room visits, and even death (Becker et al., 2007). Controlled clinical trials are labor-intensive, time-consuming, expensive and at times fail to discover possible interactions. Another critical factor is that, many drugs are prescribed (or consumed without prescription) on a daily basis for chronic ailments and it is difficult to track their consumption, making DDIs a highly impactful as well as challenging problem.

Most AI and machine learning based approaches for DDI can be categorized on basis of the type of features used (such as text-based, which involves analysis of medical abstracts and EHRs or structure-based, which involves chemical/molecular properties of drugs) or on the basis of algorithmic properties (such as supervised prediction or unsupervised clustering). **S**imilarity-based **K**ernel for **I**dentifying **D**rug-**D**rug interaction & **D**iscovery (SKID^3) framework (Dhami et al., 2018), is a hybrid approach that uses structure-based features.

The problem setting in SKID^3 (Dhami et al., 2018) differs from related approaches in the following ways– (a) As opposed to, majority of current work that leverages *information extraction* through text mining, SKID^3 *focuses on structured sources of information to discover interactions*, (b) Several approaches do utilize structural information, but in a vectorized representation which leads to loss of critical information; SKID^3 instead analyses similarities in structural patterns, and lastly, (c) Unlike most existing solutions, SKID^3 (Dhami et al., 2018) *provides a general and extensible framework that admits heterogeneous characterizations arising from varied sources.*

Thus the major contributions of the SKID^3 framework can be summarized as follows: **(I)** It identifies chemical interaction potential between two drugs, through a *reachability* measure using *knowledge-refined random walks* between them, **(II)** It provides a novel framework that combines multiple similarity measures into a unified kernel that exploits and fuses their potentials including measures that capture molecular and chemical similarities through SMILES strings and MACCS fingerprints, consequently posing the problem as a heterogenuous kernel-learning task. **(III)** It formulates the learning task as a *bilinear program*, which is a non-convex optimization problem and solves it via an alternating minimization approach, and **(IV)** Dhami et al., (2018), shows how it can identify DDIs, missing from DrugBank, but existing in other independent sources clearly demonstrating the potential for DDI discovery. The complete pipeline of SKID^3 is shown in figure 8.4.

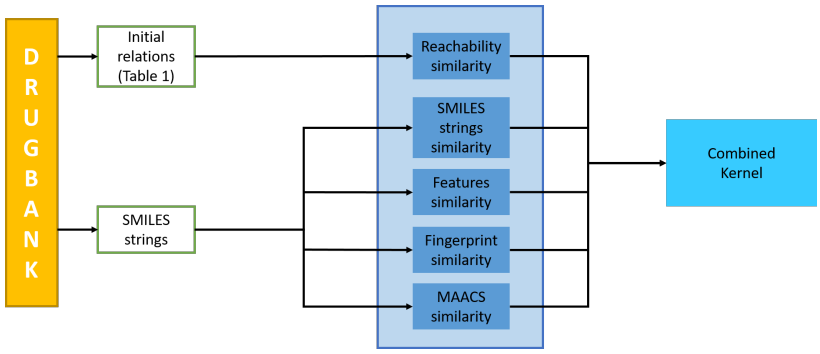


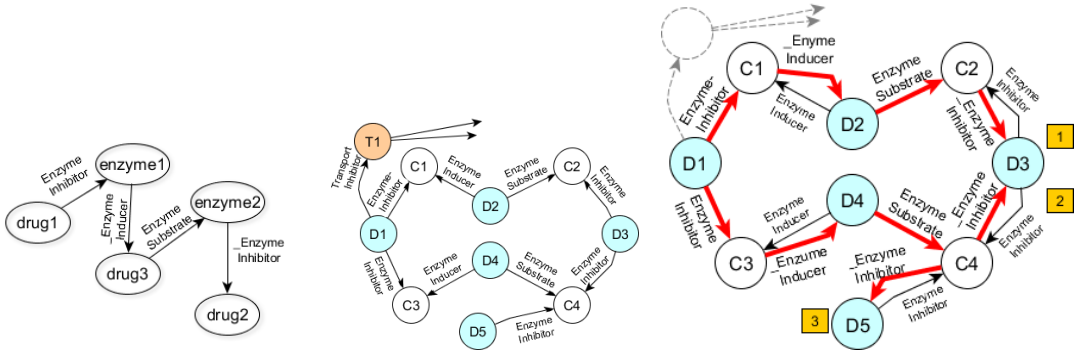
Figure 8.4: Complete pipeline for creation of SKID^3

Drug-Drug Interactions: The study of DDIs involve several biochemical entities and their relationships. For instance, the *target* or drug target is the protein modified by the drug in order to achieve the desired effect once the drug is administered to the body. *Enzymes* are catalysts that accelerate biological reactions, while *transporters* are proteins that help drugs reach the intended target (Nigam, 2015), and also help in determining whether the drug will be absorbed, distributed or eliminated. Additionally, DDIs can be either *synergistic* (positive, and help increase the effect of the drugs) or *antagonistic* (negative, cause serious side effects). August et al., (1997) categorizes

(multiple) drug metabolism as, (1) **Pharmacokinetic:** the effect that drugs goes through when administered, for example, it is absorbed or metabolized, when administered simultaneously, and (2) **Pharmacodynamic:** the effect that body goes through when a drug is administered which, in case of DDIs, implies the effect of one drug on another drug when they are operating on the same target or even different targets. Inspired by that, SKID^3 postulates that there should exist a "path of relationships" describing the molecular and structural properties of the drugs, especially when there is an interaction.

8.2.1 Kernel Learning for Drug Drug Interactions

SKID^3 (Dhami et al., 2018) formulates the DDI problem as a multiple-kernel learning task (Bach et al., 2004; Lanckriet et al., 2004), over several *heterogeneous* similarity-based kernels that are constructed from 5 different association measures, (1) The chemical reactivity path based **"Reachability"** measure, and 4 structural similarity measures (2) **Feature Similarity** (FS), (3) SMILES **String Similarity** (SS), (4) **Molecular Fingerprint** (FP) and (5) **Molecular ACCess System**. Lets us look deeper into these measures.



(a) Parameterized Walk '$\mathbb{W}$' (b) Network of reactions '$\mathcal{G}$' (c) Identified paths/instances in the graph

Figure 8.5: Instantiation process of a parameterized random walk $\mathbb{W}$ (left) is equivalent to subgraph matching for a given motif. The graph $\mathcal{G}$ (middle) shows a part of the chemical reaction network (D_x, C_x & T_x indicate drugs, enzymes and transporters resp.). The rightmost figure shows how 3 different instances/paths (marked in **red**) have been identified that satisfy $\mathbb{W}$.

Reachability

A key component describing drug-drug interactions is the charaterization of how two drugs chemically react with each other. This is captured using a *directed graph of known chemical reactions* between drugs and enzymes, transporters etc. using ADMET (absorption, distribution, metabolism, excretion and toxicity) features. The idea of *reachability* follows from the intuition that two drugs are likely to interact with one another if one is reachable from the other via one or more paths in an ADMET knowledge graph. Graph theoretic approaches for reachability analysis (Lü and Zhou, 2011; Taskar et al., 2004) are insufficient in this problem setting since ADMET networks are multi-relational, directed and relatively sparse involving several thousands of entities/nodes representing drugs, enzymes, targets etc. An iterative search within such a large graph may be intractable. Hence, for this task, Dhami et al., (2018) adapts the the path ranking algorithm (PRA) (Lao and Cohen, 2010) which has been provably successful in generating robust predictive models for relation extraction and reachability analysis. Reachability measure is obtained by first constructing a knowledge graph for known chemical reactions using the ADMET features followed by a ***Guided (Parameterized) Random Walk Generation*** (which performs walks of varying length, constrained by expert domain knowledge, on the relational schema) and instantiation (finding paths in the knowledge graph that satisfy the parameterized random walks) and finally, generating the reachability score by obtaining the cardinality (count) of the path instances for a given drug pair. Each random walk $\mathbb{W}$ is equivalent to a conjunctive First Order Logic formula. Thus the instantiation and cardinality based scoring is equivalent to satisfiability counting problem in logic which is a combinatorially hard problem ($\#P$-complete). It adapts *FACT* (Chapter 3) for efficiency.

Structural similarities measures based on SMILES and SMARTS strings

The simplified molecular-input line-entry system (SMILES) is a commonly-used specification for describing chemical and molecular structure using ASCII strings. The SMILES arbitrary target specification (SMARTS) is an extension of SMILES that is also commonly used for specifying

molecular sub-structures precisely. Four similarity measures are extracted from molecular and chemical properties of the drug (specified by SMILES and SMARTS strings),

(S1) Molecular **Feature Similarity** (FS) compares the chemical properties of two drugs using 19 features extracted from their SMILES strings.

(S2) SMILES **String Similarity** (SS) is the similarity between the SMILES strings themselves, which is calculated using edit distance between the strings.

(S3) **Molecular Fingerprint** (FP) similarity is computed between the fingerprints, which are bit-string representations of the molecular structure.

(S4) **Molecular ACCess System** (MACCS) keys are a particular type of fingerprint generated from SMARTS strings. Similarities on MACCS are commonly used in the drug discovery domain, though they have been proven to be useful on the DDI domain as well (Vilar et al., 2014).

SKID^3 addresses kernel learning at an element-wise, local and global level, enabling us to learn robust models for discovery of new drug-drug interactions.

Given: For N drugs, M interaction similarities S_m, a small subset of *known* interactions y_{ij} for pairs $ij \in \mathcal{L} \subset \mathcal{P}$,
Learn: A fused kernel $Z \succeq 0$, and combination weights $\alpha_m \geq 0$, $\sum_{m=1}^{M} \alpha_m = 1$,
Predict/Discover: Previously unknown pairwise drug-drug interactions $\hat{y}_{ij} = \mathsf{sign}(z_{ij})$, for pairs $ij \in \mathcal{U} = \mathcal{P} \setminus \mathcal{L}$.

Neighborhood-preserving Kernel Fusion

Intuitively, each interaction/similarity measure is a *graph* that provides a *different view of the neighborhood* of a drug. That is, each similarity matrix S_m represents a fully-connected graph with s_m^{ij} representing the edge weight between drugs $\mathbf{d}_i$ and $\mathbf{d}_j$. Since each S_m measures similarities differently, the neighborhood of a drug $\mathcal{N}_m(\mathbf{d}_i)$ with respect to different S_m will be different. In order to effectively incorporate this multi-view neighborhood information, we construct graph Laplacians $L_m = (1+\delta)I_N - D^{-\frac{1}{2}} S_m D^{-\frac{1}{2}}$, $m = 1, \ldots, M$, for each similarity, where I_N is an $N \times N$

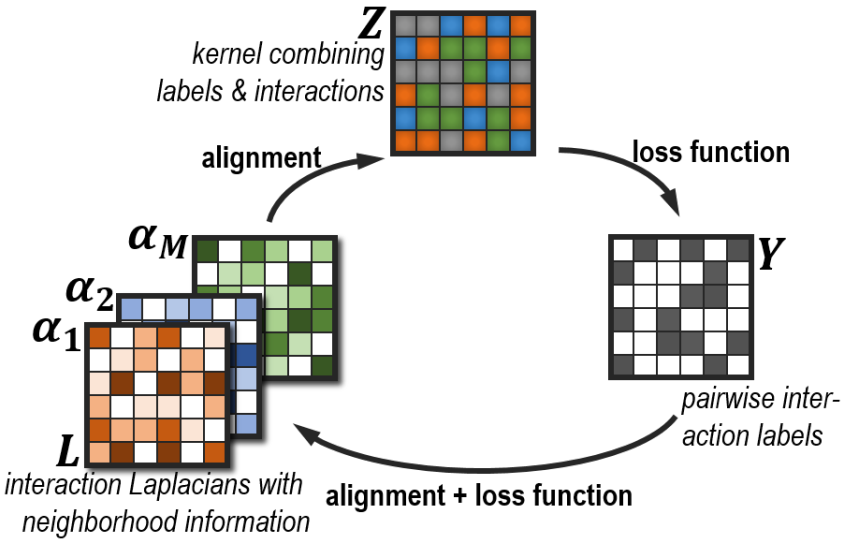


Figure 8.6: We learn a positive semi-definite kernel Z and combination weights α_m for various similarity measures. These similarities represent interaction scores, which help determine how likely two drugs are to interact. The similarity measures are expressed through Laplacians, which view the interactions as a *neighborhood graph*. In this manner, we can incorporate local information into the kernel. Z is element-wise consistent with the labels.

identity matrix and D is a diagonal matrix with entries $d_{ii} = \sum_{j=1}^{N} s_{ij}^m$ (the row sum of the similarity matrix S_m). The target is to learn a kernel that fuses neighborhood information $\mathcal{N}_m$ from multiple interaction types,

SKID^3 formulates the following kernel learning problem that maximizes the decision margin:

$$\begin{aligned} \underset{L,\, Z,\, \boldsymbol{\alpha}}{\text{minimize}} \quad & \overbrace{\langle L,\, Z\rangle + \langle L,\, Y\rangle}^{\text{alignment}} + \overbrace{\lambda_1 r(\alpha)}^{\text{regularization}} \\ & \underbrace{\lambda_2 \left(\ell_1(Z,\, Y) + \ell_2(L,\, Y)\right)}_{\text{loss functions}} \\ \text{subject to} \quad & L = \sum_{m=1}^{M} \alpha_m L_m,\ \boldsymbol{\alpha} \geq 0,\ \mathbf{e}'\boldsymbol{\alpha} = 1,\ Z \succeq 0. \end{aligned} \tag{8.1}$$

The variable $L = \sum_{m=1}^{M} \alpha_m L_m$ is a *convex combination* of the Laplacians L_m. The *alignment* terms imply alignment-based regularization for kernel learning (Cortes et al., 2012) and incorporates local neighborhood information encoded in the different Laplacians as well as the labels into learning $\boldsymbol{\alpha}$ (the combination weights) and Z (the fused kernel, s.t. $Z \succeq 0$). Specifically, $\langle L,\, Y\rangle$

encourages the weights on the Laplacians α to be consistent with the labels Y. The impact of labels is also propagated into Z by the $\langle Z, Y \rangle$ term. The entries of the fused kernel $z_{ij} \in Z$ directly provide a unified interaction score for prediction, $\hat{y}_{ij} = \mathsf{sign}(z_{ij})$. To maximize *interaction margin* is maximized a *hinge loss* (for ℓ_1 in Equation 8.1) is used to ensure that $y_{ij} z_{ij} \geq 1$ holds. A L_2 regularizer is used over the combination weights, $r(\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\alpha}\|_2^2$.

The learning problem is solved using *alternating minimization* (Csiszár and Tusnády, 1984) between the $\boldsymbol{\alpha}$ and Z. Both sub-problems of alternating minimization were solved using `SDPT3` (Toh et al., 1999). For further details refer to Dhami et al., (2018)

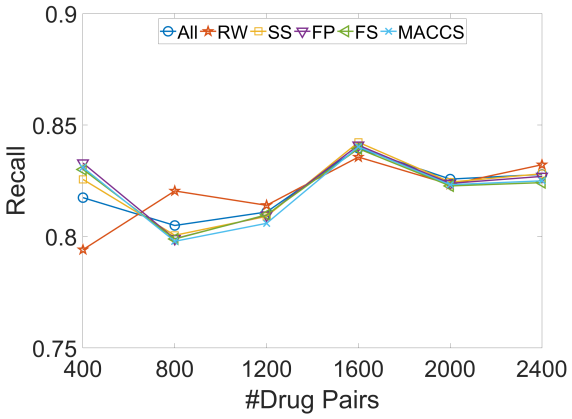
8.2.2 Evaluation

Evaluations are conducted with a data set of 78 drugs obtained from DrugBank[1]. This gives rise to 3003 possible interactions[2] in which 603 drug pairs were kept as the holdout test set. Different methods were trained with increasing number of drug pairs ranging from 400 to 2400, chosen randomly for each run.

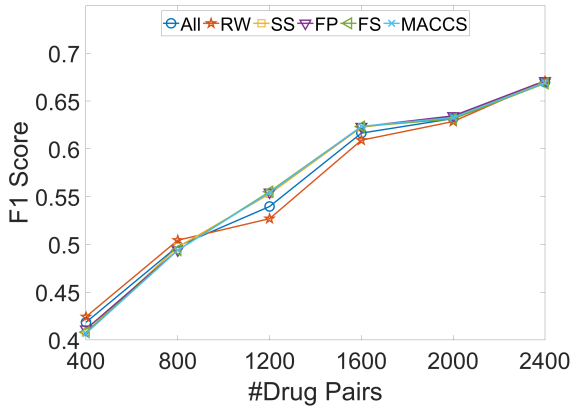
Effectiveness: Figure 8.7(b) shows that the individual kernels learned from each individual similarity measure (described in Sec. 8.2.1) are able to perform reasonably well on the DDI prediction task. However, our observations also confirm our hypothesis that a fused single kernel over molecular structural similarities and chemical reaction pathways could combine the advantages of of both showing a more stable performance. Figure 8.7(c) shows the change of the learned weights as the number of training drug pairs increases. A key observation is that the influence of the random walk (RW) similarity decreases, while the weight of the molecular structure similarities increases with increasing training pairs. This suggests that RW similarities are particularly effective in smaller

[1] `https://www.drugbank.ca/`

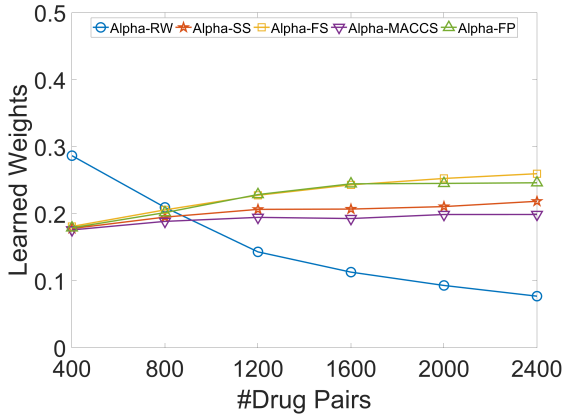
[2] Given n drugs, there will be a total of $\binom{n}{2} = \frac{n(n-1)}{2}$ interactions.



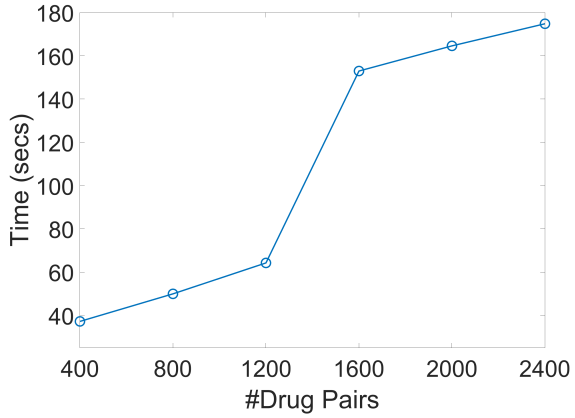
(a) Recall, averaged over 5 runs on a hold-out test set



(b) F_1-score, averaged over 5 runs on a hold-out test set



(c) Similarity matrix weights, $\boldsymbol{\alpha}$



(d) Training Time for learning the *combined* kernel, SKID^3

Figure 8.7: Experimental results, with kernels learned using **All** similarity measures (SKID^3), random-walk reachability (**RW**), SMILES string similarity (**SS**), molecular feature similarity (**FS**), molecular fingerprint similarity (**FP**) and MACCS fingerprint similarity (**MACCS**). (a)–(b) Classification performance of each kernel with increasing training sets; (e) Change in weights of each individual kernel; (f) Training time for learning the combined kernel (SKID^3).

databases, for targeted identification of interactions. This could have a positive impact on the challenges of sparse low-quality data in treatment planning (Chapter 1, Section 1.2, Example 3).

Discovery: Another key objective in this work was to see if SKID^3 is able to *discover new interactions* not annotated in DrugBank. Table 8.2 presents *a few drug pairs* that are supposedly "incorrectly classified" by our method using the DrugBank ground truth. Instance of Drugbank

database dated April 2017 (used as ground truth) shows absence of such interactions whereas in the instance dated February 2018 they were added. Table 8.2 also shows that the interactions discovered by our approach can be supported by independent sources or research proving the potential for SKID^3 for discovery. This is crucial for drug surveillance database updates as well as effective treatment planning with in our envisioned HAAI framework.

Table 8.2: Table depicting example drug pairs where the prediction does not match the ground truth (DrugBank). However, we additionally cite sources (last column) that support our prediction.

Drug 1	Drug 2	Ground Truth	Predicted Class	Independent Source
Amitriptyline	Tamsulosin	Not interacting	Interacting	Drugs.com (Multiple, a)
Omeprazole	Metformin	Not interacting	Interacting	Nies et al. (Nies et al., 2011), rxlist.com (Multiple, c)
Salbutamol	Clonidine	Not interacting	Interacting	Thoolen et al. (Thoolen et al., 1984)
Cephalexin	Diclofenac	Not interacting	Interacting	Ali et al. (Ali et al., 2015)
Amoxicillin	Metronidazole	Not interacting	Interacting	Pavicic et al. (Pavicić et al., 1991)
Amphetamine	Salbutamol	Not interacting	Interacting	DrugBank[??]
Cephalexin	Methadone	Interacting	Not interacting	Drugs.com (Multiple, b)

Finally, Figure 8.7(d) shows the time taken by our method. The training time increases linearly with the number of drug pairs, making it potentially scalable. Possible challenges in scaling, due to the hard problem of instantiation in reachability metric, can be overcome by our approximation techniques (*FACT* and *MACH*) outlined in Chapters 3 & 4.

8.3 Boosted Arithmetic Circuits

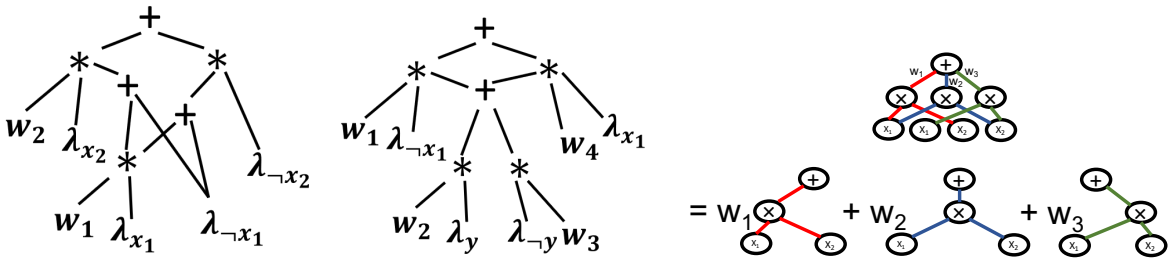
WE have discussed in earlier chapters how, probabilistic inference can become intractable in large parameter spaces. This is especially critical in inference tasks with SRL models. Lifted inference techniques (Poole, 2003; de Salvo Braz et al., 2005; Milch et al., 2008; Kersting et al., 2009; Singla and Domingos, 2008) attempt to address this challenge by exploiting symmetry and grouping indistinguishable patterns and inferring over such groups as a whole. However, as the amount of evidence grows a probabilistic graphical model becomes increasingly asymmetric. While approximate lifting can alleviate the need for exact symmetries it is an open research area. Another perspective of tractable inference is via compilation to alternate structures (such as Arithmetic Circuits (Darwiche, 2003)) which facilitate tractable inference (complexity is polynomial in the

size of the parameter space). Consequently, Arithmetic Circuits (AC) and Sum-Product Networks (SPN) have recently gained significant interest by virtue of being tractable deep probabilistic models. Most previous work on learning AC structures, however, hinges on inducing a tree-structured AC and, hence, may potentially break loops that may exist in the true generative model. To repair such broken loops, Ramanan et al., (2019) propose a novel gradient-boosted method for structure learning of discriminative ACs (DACs), called DACBOOST. Since, in discrete domains, ACs are essentially equivalent to mixtures of trees, DACBOOST decomposes a large AC into smaller tree-structured ACs and learns them in a sequential, additive manner. The resulting non-parametric manner of learning the DACs results in a model with very few tuning parameters making the learned model significantly more efficient (as demonstrated via the empirical evalutions).

Rooshenas and Lowd (Rooshenas and Lowd, 2016), proposed a discriminative learning method for ACs that greedily searches through the space of features. While successful, their work has two limitations: (1) a large number of parameters that must be tuned and (2) the ACs are typically limited to being tree-structured and, hence, may break loops. Inspired by the intuition that many weak learners, in our case tree-structured ACs, could be more successful in learning a conditional distribution, DACBOOST (Ramanan et al., 2019) proposes a non-parametric learning method for discriminative ACs (DACs) based on gradient-boosting.

8.3.1 Arithmetic Circuits

In traditional probabilistic graphical models inference is a function of the tree-width and hence complex (Chandrasekaran et al., 2008). The problem of tractability in the presence of large amounts of evidence has been explored from multiple directions. One of them is via compiling models into representations suitable for efficient inference such as (deep) probabilistic architectures like Arithmetic Circuits (ACs) (Darwiche, 2003; Lowd and Domingos, 2008; Lowd and Rooshenas, 2013) and Sum-Product Networks (SPNs) (Poon and Domingos, 2011; Gens and



(a) Arithmetic Circuit (AC) (b) Conditional AC (c) Decomposable AC, as a mixture of trees.

Figure 8.8: **Left:** Example arithmetic circuit that represents a markov random field over two variables x_1 and x_2 and having potentials w_1 and w_2 for the 2 features $x_1 \wedge x_2$ and x_2. **Middle:** Conditional AC that represents the distribution $P(y|x_1)$ where y is a query variable and x_1 is an evidence variable. **Right:** A complete and decomposable AC, here actually an SPN, is a mixture of trees (Zhao et al., 2016).

Domingos, 2013; Rooshenas and Lowd, 2014). The probability distribution induced by a probabilistic graphical model can be represented by a multilinear function called a *network polynomial* that consists of sum of product of indicator variables λ and the parameters w of the network (Darwiche, 2003). ACs can be used to compactly represent the network polynomial.

An *Arithmetic Circuit* $AC(\mathcal{X})$ is a rooted, directed acyclic graph over the variables $\mathcal{X}$. It contains $+$ or $*$ as internal nodes and its leaf nodes are labeled with either a non negative parameter w or an indicator variable λ. Then for any instantiation x, the value of the circuit $AC(\mathcal{X})$ is computed by assigning indicator λ_x the value 1 if $\mathcal{X}$ is compatible with instantiation x and 0 otherwise. Figure 8.8(a) shows an AC over two binary random variables x_1 and x_2 which is are generative and hence model a joint distribution by definition (Figure 8.8(b) shows a conditional AC). The network polynomial for the AC, which is multilinear in the λ and w variables, can be written as; $\lambda_{x_1}\lambda_{x_2}w_1w_2 + \lambda_{x_1}\lambda_{\neg_{x_2}} + \lambda_{\neg_{x_1}}\lambda_{x_2}w_2 + \lambda_{\neg_{x_1}}\lambda_{\neg_{x_2}}$. The complexity of evaluating an AC is linear in the size of the circuit. The two properties that lead to tractable ACs are *Decomposibility* and *smoothness*, as discussed in (Darwiche, 2003; Lowd and Rooshenas, 2013). Zhao *et al.* (2016) have shown that ACs can be viewed as mixtures of trees as illustrated in Fig. 8.8(c).

8.3.2 Boosted Learning of Discriminative ACs

In most prediction tasks, we are interested in computing the conditional distribution of a predetermined set of response variables given their parents. Thus, modeling such a conditional via a DAC, where we can treat the entire set of parents as evidence variables, is sufficient. Rooshenas and Lowd (Rooshenas and Lowd, 2016) proposed DACLEARN and showed how it can learn the structure of a DAC by optimizing the penalized conditional log-likelihood (CLL) and is considerably more tractable even for large tree-width models, as opposed to generative training of a typical AC.

Functional Gradient Boosting (FGB) (Friedman, 2001) postulates that lea111rning conditional probabilistic models can be transformed into learning a sequence of function approximation problems such that a conditional probability distribution can be represented as a weighted sum of regression models learned sequentially via a stage-wise optimization (Dietterich et al., 2004; Natarajan et al., 2012; Khot et al., 2011). Thus for a particular example y_i its conditional distribution given its parents $\mathbf{x}_i$ can be learned by fitting a model $P(y|\mathbf{x}) \propto e^{\psi(y,\mathbf{x})}$, and can be expressed non-parametrically in terms of a potential function $\psi(y_i; \mathbf{x}_i)$: $P(y_i \,|\mathbf{x}_i) = \frac{e^{\psi(y_i;\mathbf{x}_i)}}{\sum_{y'} e^{\psi(y';\mathbf{x}_i)}}$. Instead of learning in the parameter space (P), gradient can be obtained in the functional-space ψ and successively approximate ψ as a sum of weak learners, which are typically regression trees. The functional gradient after m iterations is expressed as,

$$\Delta_m = \eta_m \cdot E_{\mathbf{x},y}\left[\frac{\partial}{\partial \psi_{m-1}} \log\, P(y \mid \mathbf{x};\, \psi_{m-1})\right]$$

where η_m is the learning rate. Fitting a regression tree h_m on the training examples $[(x_i, y_i), \Delta_m(y_i; x_i)]$ is a close and reasonable approximation of the desired Δ_m. Ramanan et al., (2019) proposes a learning algorithm based on gradient-boosting for full model learning (structure + parameter learning) of DACs (Fig. 8.9).

Given a data set $\mathcal{D}(\mathcal{X}, \mathcal{Y})$, where $\mathcal{Y}$ is a set of query variables, and $\mathcal{X}$ is a set of evidence variables, **find** the structure and parameters of a discriminative arithmetic circuit (DAC), i.e., the distribution $P(\mathcal{Y}|\mathcal{X})$).

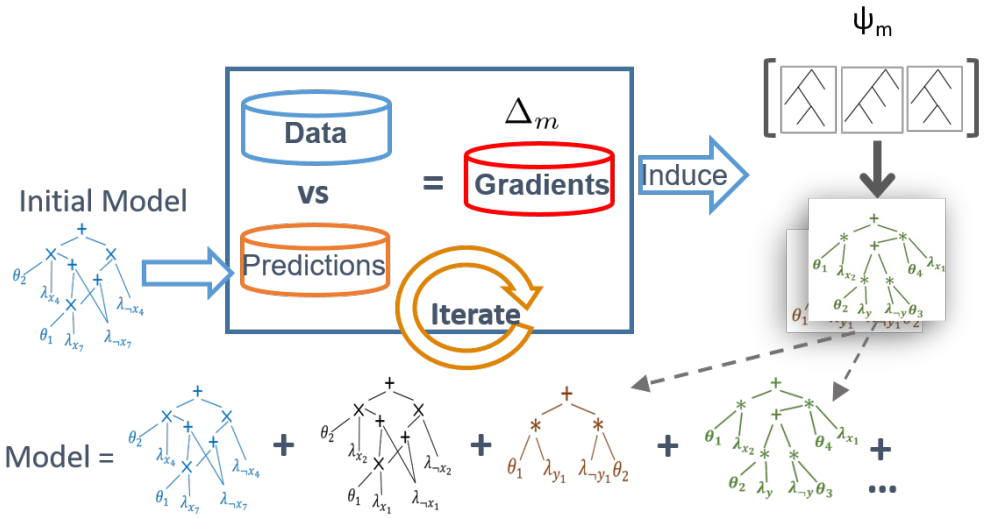


Figure 8.9: Learning of Discriminative Arithmetic Circuits via boosting. As can be seen, at each iteration a small weak DAC is learned for each query variable. Once a DAC is learned at each iteration, the weights of the examples are computed and a new AC is learned. These are then added to the model and the process continues until convergence.

Gradient Boosting for Conditional ACs

Following DACLearn (Lowd et al. 2016) that optimizes conditional log-likelihood of train data set $\mathcal{D}$ by finding the best set of features f, where, $\mathbf{y} = \mathcal{Y}^{(p)} \subseteq \mathcal{Y}$ and $\mathbf{x} \subseteq \mathcal{X}$, $CLL(\mathcal{D})$ is defined as, $CLL(\mathcal{D}) = \sum_{(\mathbf{y},\mathbf{x})\in\mathcal{D}} \log P(\mathbf{y}|\mathbf{x})$ Since functional-gradient boosting obtains point-wise gradient for each example separately the log-likelihood for a specific example i and a specific query variable y (we drop the superscript for brevity),

$$\begin{aligned}\log P(y_i = 1|\mathbf{x}_i) &= \sum_j w^j f^j(y_i = 1|\mathbf{x}_i) - \log Z(\mathbf{x}_i) \\ &= \sum_j w^j f^j(y_i = 1|\mathbf{x}_i) - \log \sum_{y'} \exp\Big(\sum_j w^j f^j(y_i = y'|\mathbf{x}_i)\Big)\end{aligned}$$

where Z denotes the normalization constant (partition function), i denotes the i^{th} example, j denotes the j^{th} feature of the evidence set and (p) denotes the p^{th} query variable. Like CRFs, the definition of DACs, the functional representation. ψ for i-th example is directly the potential,

$\psi(y_i|\mathbf{x}_i) = \sum_j w^j f^j$. Thus the pointwise gradient of CLL w.r.t ψ $\forall i$ is;

$$\frac{\partial P(y_i|\mathbf{x}_i)}{\partial \psi(y_i|\mathbf{x}_i)} = I(y_i|\mathbf{x}_i) - \frac{\exp(\psi(y_i = 1|\mathbf{x}_i))}{\sum_{y'} \exp(\psi(y_i = y'|\mathbf{x}_i))} = I(y_i|\mathbf{x}_i) - P(y_i = 1|\mathbf{x}_i)$$

At every iteration a weak learner (DAC in our case) is learned to fit these pointwise gradients due to the previous iteration. The score of the structure using the set of candidate features $\mathcal{F}$ can be rewritten as, $score(\mathcal{F}) = \Delta_{cll}(\mathcal{F})$, where Δ_{cll} denotes the change in CLL. The goal is to identify the features that maximize this change in CLL. Explicit model complexity penalty is not necessary since learning weak models automatically takes care of regularization by controlling the depth of the learned ACs.

At any given iteration DACBOOST learns a small, tree-structured AC by searching through the space of potential features to add next by minimizing the weighted variance of the conditional distribution according to the current model. Once the AC is constructed the weights w^j at the leaves of the DAC are estimated by maximizing the increment in CLL function:

$$\Delta_{cll}(\mathcal{F}) = \sum_{(\mathbf{y},\mathbf{x}) \in D} \sum_j w^j \hat{P}(f^j|\mathbf{x}) - \Delta \log Z(\mathbf{x})$$

where $\hat{P}$ is the new empirical probability distribution after introducing the new candidate features. Figure 8.9 presents the overall architecture for our DACBOOST approach. Please refer to Ramanan et al.,(2019) for a detailed discussion of the algorithm. Note that, While DACLEARN induces tree structured ACs, DACBOOST is capable of learning DAGs.

8.3.3 Evaluation

DACBOOST is evaluated on four novel clinical/medical data sets, a network traffic for DDoS attack detection data set and some standard data sets, against the state-of-the-art discriminative structure learning algorithm for ACs, *DACLearn*.

The novel domains include - **Alzheimer's**: The Alzheimer's Disease NeuroIntiative (ADNI[3]) which is designed to verify whether MRI and PET images, genetics, cognitive tests and blood

[3] www.loni.ucla.edu/ADNI

biomarkers can be used for early prediction Alzheimers disease; **Drug-Drug interactions (DDI)**, which consists of 78 drugs and pairwise interactions based on chemical reaction pathways (Dhami et al., 2018); **DDoS attack detection (DDoS)**: which was employed in the work of Ricks et al. (Ricks et al., 2018), benign and large-scale botnet network traffic is captured for use in DDoS attack detection and our goal is to learn a CAC model for attack modeling given benign and botnet network traffic. **Post-Partum Depression (PPD)**: inspired from Natarajan et al (Natarajan et al., 2017), the goal is to model post-partum depression diagnosis based on online questionnaire data including demographics, family history (relationship), social support, economic status, infant behavior and CDC questions; and finally, **Parkinson's**: Parkinson's Progression Markers Initiative (PPMI[4] (Dhami et al., 2017)) is a study designed to identify biomarkers that impact Parkinson's progression in a subject. In addition, the benchmark data sets included the ones that were extensively used in prior work on learning SPNs and ACs (Rahman et al., 2014; Davis and Domingos, 2010; Gens and Domingos, 2013; Rooshenas and Lowd, 2014, 2016).

Effectiveness and Efficiency: The predictive performances and running times on the 5 novel data sets are presented in Table 8.3. As can be seen, it is fairly clear that DACBOOST is at least as good as or better than DACLearn in several of these data sets (4/5) demonstrating the effectiveness of DACBOOST . The lower performance observed in `DDI` data set is possibly caused by sparsity. The values w.r.t. several features were 0 uniformly across all the examples rendering them useless for constructing weak ACs and hence boosting does not perform as well as DACLearn (though not significantly worse). In `DDoS` data set, large number of repeated examples paired with subsampling of negatives in DACBOOST may have resulting in slower convergence.

Results on benchmarks 8.4, leads to a key observation that DACBOOST is significantly faster than the state-of-the-art in most cases. In some domains such as `MSNBC` and `Plants`, the learning

[4] `www.loni.ucla.edu/PPMI`

Table 8.3: Evaluation results of DACBOOST on real domains (data sets). Conditional log-likelihood, *CLL*, illustrates the effectiveness (higher ⇒ better), while the running time, *Time(Sec.)* shows the efficiency (lower ⇒ better). *Speedup* is the ratio of the running time of DACLearn to that of DACBOOST.

Data sets	DACBOOST		DACLearn		Speedup
	CLL	Time (Sec.)	CLL	Time (Sec.)	
ADNI	**-0.178**	**0.950**	-0.180	1.90	2.000
DDI	-0.264	**133.87**	**-0.245**	158.86	1.186
DDoS	**-0.018**	133.52	-0.019	**121.11**	0.907
PPD	**-0.411**	**12.19**	-0.785	13.83	1.134
PPMI	**-0.163**	**353.74**	-0.188	522.99	1.478

Table 8.4: Results of DACBOOST on benchmark data sets that have been used and reported in DACLEARN (Rooshenas and Lowd, 2016). Conditional log-likelihood, *CLL*, shows the effectiveness (higher ⇒ better), while the running time, *Time(Sec.)* shows the efficiency (lower ⇒ better). *Speedup* is the ratio of the running time of DACLearn to that of DACBOOST.

Data sets	DACBOOST		DACLearn		Speedup
	CLL	Time (Sec.)	CLL	Time (Sec.)	
Jester	**-0.409**	**481.06**	-0.665	9851.41	20.478
KDDCup	**-0.073**	**756.68**	**-0.073**	1248.77	1.650
Kosarek	**-0.011**	**1355.84**	-0.021	3778.92	2.787
MSNBC	**-0.108**	**76.61**	-0.110	12153.90	158.640
NLTCS	**-0.148**	**17.01**	-0.152	40.35	2.372
Plants	-0.298	**467.60**	**-0.255**	13077.74	27.967
WebKB	**-0.178**	**6154.40**	-0.293	17840.81	2.900

time of DACBOOST is order-of-magnitude smaller. Thus boosted learning of tractable models such as DACs will lead to efficient learning.

In summary, the empirical evaluations on novel and established benchmark data sets appear to clearly demonstrate the potential for learning DACs in a stage-wise manner. Even in domains where there is a small loss in performance, the learning time is significantly faster than learning a full DAC. Hence, boosted deep tractable models paired with our approximation techniques could provide new avenues for scalable learning with dense data in structured spaces.

8.4 Adaptations of Count Approximation Frameworks

Our novel approaches, outlined in the earlier chapters, to address the key challenges considered by this dissertation have also been successfully adapted in related research.

8.4.1 Relational Restricted Boltzmann Machines

Our approximate counting technique, have been successfully adapted for efficient learning/inference in relational deep models specifically in Relation Restricted Boltzmann Machines (RRBM) (Kaur et al., 2017) for counting inside relational random walks. Restricted Boltzmann Machines (Ackley et al., 1985) provide us with the key principle behind deep architectures. Like other techniques in its class, it uses vectorized feature space representation of the domain.

Consequently, Kaur et al., (2017) extended it to the relational setting by incorporating parameterized features. Such features are constructed by performing "lifted" (relational) random walks on the data. The key idea is that, if the random walk discovers a path between one entity to another in a knowledge graph, then that path could indicate existence some target relationship between those two entities. Thus, in a classification setting, a relational random walk can be posed as a definite clause whose body is the generalized pattern discovered by the walk, and the head is the target relation between the two entities between whom the walk was performed. *For instance, in the example MLN for our hospital resource forecasting scenario in Chapter 2, the body of the clause C_1 can be discovered by walking on the relational schema going between the hospital entity and disease entity going via* `<Neighborhood:Nb()>`, `<In-Neighborhood:In()>`, `<Contact()>` *to a* `Confirmed` *case of disease* d. The importance or contribution of a lifted walk in the final predictive task is a matter of parameter estimation (akin to the weights in MLN).

In RRBM, instead of explicitly computing the weights of walks and including them in the final model in a greedy fashion, they are passed as input features into a Restricted Boltzmann Machine (RBM). However, there is a key step involved in this approach. RBM's input layer is designed

to work with vectorized input. Relational features are incorporated by **propositionalizing**, *i.e.* passing the count vector where each element of the vector is the satisfiability count of a relational random walk. If N lifted random walks are generated (by an extended version of PRA (Lao and Cohen, 2010)), then the input vector is of size N, where value at position i is the satisfiability count of the random walk R_i/θ (θ is the partial substitution for target entities similar to Chapter 2, Section 2.2): $\#[R_i/\theta]$. Now we are already aware that computing such counts in an exact fashion is hard (#P-complete). Hence the authors use our count approximation strategies. The reported results in this work are with respect to *FACT* since all tasks involved relationships of at most binary arity. However, for any task that must characterize relationships of arbitrary arity (such as $\texttt{LabTest}(\texttt{person}, \texttt{hospital}, \texttt{testname})$), *MACH* can be seamlessly integrated into the propositionalization step for count vector approximation.

8.4.2 Drug-Drug Interaction and Discovery

Our count approximation techniques have also been adapted for efficient predictive modeling in a real-world pharmaco-vigilance task in one of our recent work (Dhami et al., 2018). This work has been described in detail earlier in Section 8.2. This work aims to predict/discover drug pairs that are likely to react with each other (possibly adversely) when taken together. Our model used a combination of several kernels, one of which is a reachability kernel that represents likelihood of reactivity between twp given drugs based on chemical pathways. This requires counting such pathways in chemical reaction networks involving ADMET features such as enzymes, transporters and so on. Such pathways, similar to the RRBM case described earlier, are discovered by performing lifted random walks based on PRA (Lao and Cohen, 2010). To pose it as a metric for kernel construction, satisfiability counts are computed. The intuition is that, we are trying to estimate the actual number of ground paths in the ADMET knowledge graph that satisfy the 'shape' given by a lifted random walk. Naturally for efficient learning our techniques (*FACT* and *MACH*) were used for approximating these counts.

8.4.3 Relational Logistic Regression

Our approximate counting technique, have been successfully adapted for efficient learning/inference in several probabilistic logic models including structure learning in Relational Logistic Regression (Ramanan et al., 2018) which considers the problem of learning a directed SRL model of **Relational Logistic Regression** (RLR) (Kazemi et al., 2014; Fatemi et al., 2016). One of the key advantages of RLR is that they scale well with population size unlike other methods such as MLNs (Poon et al., 2008) and hence can potentially be used as a powerful modeling tool for many tasks. Th population invariance property of RLRs arise due to the fact that they factor in the negative (unsatisfied) instances of clauses in their formulation. As we will see later this makes RLRs an interesting case for adaptation of our count approximation techniques.

A straightforward solution to learning these models could be to learn the rules separately using a logic learner and then employ the parameter learning strategies (Huynh and Mooney, 2008). While reasonably easy to implement, the key issue is that the disconnect between rule and parameter learning can result in poor predictive performance as shown repeatedly in the literature (Natarajan et al., 2012; Richardson and Domingos, 2006). Inspired by the success of non-parametric learning methods for SRL models, Ramanan et al., (2018) propose a gradient boosted learning method for full model learning of RLR models.

Let us take a brief look at the formulation of RLR due to Ramanan et al., (2014). For a query vector $Q(X\mathbb{X})$ it probability given its parents is given by,

$$P(Q(\mathbf{X}) = 1 \mid J) = \sigma\left(w_0 + \sum_{f, w_T, w_F \in F} w_T \cdot n_T(f/\theta_J) + w_F \cdot n_F(f/\theta_J)\right) \tag{8.2}$$

where n_T and n_F are counts with respect to satisfied and unsatisfied groundings of formula f and w_T and w_F are the respective weights. θ_J, similar to earlier definitions, indicate partial grounding of the formula f with respect to the example J. Naturally for efficient computation n_T is approximated using our techniques. However, one important aspect here is the use of n_F that renders RLRs population invariant. Counting unsatisfied groundings logically is a challenging tasks. With

our (hyper)graph based techniques it becomes comparatively easier. Counting unstatisfied groundings is is equivalent to enumerating non-existent edges or existing egdes that is not connected to the entities that does not satisfy the clause. Hence n_F can be transformed as follows,

$$n_F = n(V_f) - n_T$$

where V is the set of all logical variables in formula f. $n(V_f)$ is thus the Cartesian product of the domain sizes of all the logical variables in f ($n(V_f) = \prod_{v \in V_f} \#[v]$). Clearly $\#[v]$ can be obtained directly from the summary statistics we estimate both in FACT and MACH. Approximate value of n_T is also computed by out methods.

CHAPTER 9

CONCLUSION

Developing learning and decision making agents which are robust to both observation quality as well as density is a challenging task. We have viewed this problem as conjunction of several sub-tasks that needs to be addressed to achieve our vision. We identified that statisfiability counting, being a hard problem, as a key factor in scalability issues with dense observations in SRL. In Chapters 3 and 4 we presented our (hyper)graph-based formalisms for approximating such counts. Empirical evaluations on several benchmark domains validated some of our primary intuitions:

a.) Counting indeed affects efficiency (Poyrekar et al., 2014), and fast approximation can lead to promising scalable SRL systems capable of handling web-scale data and efficient function approximation for sequential decision making in structured noisy environments.

b.) In dense data sets, approximation error has *no statistically significant effect on estimated values* (marginals, posteriors, likelihood scores etc.).

One key advantage of our approach is that we build it as a pluggable module and can be applied on any SRL formalism that uses counting. It may be possible to scale such systems even further by integrating our approximation with model reduction techniques (Shavlik and Natarajan, 2009).

To address the challenge of robustness to observation quality, we envisioned a Human-Allied AI framework, where a human and an AI agent would solve problems together, interchange ideas in order learn and develop together. With such a framework, we aim to exploit the idea of *asymmetry of knowledge*, such that the agent can generalize better in low quality observations. Knowledge-based learning and acting have been well studied in existing literature and they show us how additional knowledge can provide an inductive bias which enables better modeling in presence of systematic noise. Our vision of human-AI collaboration transcends beyond just knowledge-based systems. We wish to establish bidirectional interaction and exchange of ideas between the

human and the agent across varied levels of domain representation. To develop this framework, we identified several sub-problems and proposed novel solutions such as KCLN, PGPLANNER and GOCI through Chapters 5, 6 and 7 respectively. We have clearly observed their advantages and potential impact compared to related alternate formalisms. For instance, PGPLANNER has shown significant positive outcome in terms of effectiveness, efficiency as well as advice complexity. GOCI has also demonstrated its effectiveness in inducing novel concepts from extremely sparse descriptions/demonstrations provided by a human collaborator. While this makes a substantial contribution towards our goal, some sub-problems still remain to be addressed. We outline them along with our ideas on potential solution approaches in the following section.

9.1 Future Directions

9.1.1 Guarantees of Count Approximation

We have presented principled formulations for count approximation in Chapters 3 and 4 based on (hyper)graph summary statistics with empirical validation of its effectiveness in scaling SRL on several benchmark domains. However, theoretical guarantees on the approximation bounds remains an open problem. We will investigate this through a PAC analysis approach (Valiant, 1984) to obtain tight error margins as a function of the size of data. We have already observed from our empirical evaluations how our approximation causes no change in performance in reasonably large samples. Thus PAC bounds on the error dependent on the sample complexity (size) will help corroborate our findings.

An alternate way to approach this challenge would be by analyzing the variance of the probability distribution of a given clause, since the expected count decomposes into the clause distribution and the Cartesian product. Chung et al., (2017) shows us how for a restricted class of unimodal distributions, variance bounds can be estimated as a function of differential entropy.

$$\frac{e^{2h(p)}}{2\pi e} \leq Var(X) \leq \frac{ce^{2h(p)}}{2\pi e}$$

for some constant $c \geq 1$, where $Var(X)$ is the variance of random variable X, $p(x)$ is its probability density and $h(p)$ is the differential entropy of $p(x)$.

Investigating, if the clause distribution can be mapped into the class of unimodal distributions outlined by Chung et al. (2017), under certain assumptions, allowing us to propose a variance bound, is an interesting future research direction.

Algorithmic and Empirical extensions: There are several possible directions to extend our work - first is more rigorous evaluation on massive data sets across other models. Second is that is while in our work we consider only conjunctions, extending the idea to count over arbitrary clauses. Finally, combining our method with grounding reduction methods such as FROG (Shavlik and Natarajan, 2009) that eliminates trivial groundings of clauses can potentially allow our algorithm to be employed in real web-scale tasks. Finally, a comparison to theta-subsumption (Maloberti and Sebag, 2004) could be interesting.

9.1.2 Generating explanations

Explainable decision-making/learning one of most critical aspects of *closing the loop* notion of intelligent agents. While there has been extensive work on the human-to-agent aspect of collaborative learning decision making, the reverse aspect of agent-to-human communication (for knowledge sharing or inconsequential dialogue) have been studied from a relatively narrow perspective. Active learning and active advice/preference elicitation frameworks (Odom and Natarajan, 2018; Das et al., 2018a) focus on generating the most useful set of queries for acquiring labels or knowledge, mixed-initiative planning works a negotiation about sub-goals and control sharing between the human and the agent. But true collaboration is subject to more effective and natural communication from the agent to the human.

Example 15. *Consider the hospital resource planning scenario presented in Example 1 and Figure 7.1 in Chapters 1 and 7 respectively. Note how in the hospital administration case, it is*

necessary for the agent to explain its inference and subsequent suggested action of more resources for the ID ward. Without the explanation offered by the agent, the administrators and physicians or the administrators may not realize the critical nature of the situation. In that case the human collaborators may not think of suggesting or teaching the agent the concepts of 'divert'.

Problem Definition

We aim to build a robust explanation framework which allows for explaining why certain decisions, suggestions, queries etc. were made as well as the reason behind specific inferred values. As a research question this is not new and some of the early explanation based systems were adapted for real-world rule-based clinical assistant AI systems (Shortliffe, 1974b). Such systems, however was restricted to exposing to the human feature values and the logic programs that were used during inference. More recent literature (Elizalde et al., 2007, 2008; Callaway and Lester, 1997; Molineaux and Aha, 2015, 2014; Chakraborti et al., 2018) study different approaches for Explanation generation in both learning and sequential decision making and planning contexts. Most of these approaches adhere to one of the following fundamental routes,

(A.) Reasoning paths over graph of explanation templates

(B.) Discourse planning in a state space of explanations

(C.) Expose change in state variables of factored MDPs.

We aim to consider an explanation framework that takes step further and provides explanations - (1) at multiple levels of generality, (2) as partial plans when needed and (3) traditional forms such as change in environment variables. Additionally, the notion of *learning-to-explain*, posed as a decision making problem in the explanation space and learn a policy in order to generate the most optimal explanation is an extremely interesting and promising further research direction.

Given: Learning or sequential decision making agent(s) A, Human collaborator(s) H, Predictive or decision-making model M

To Do: A should generate explanation(s) for (1) The model itself or (2) about a decision (or marginal/MAP of query variable) given the model. The explanations should be at appropriate level of generality. Finally, A must **learn to generate the best explanation** that captures appropriate generality, is optimally comprehensible to humans and justifies the decision/prediction.

Explanations with Hierarchical Decision Making Systems

We aim to build upon our earlier work, PGPLANNER (Das et al., 2018a), for active preference elicitation in hierarchical planning. Hierarchical planners succinctly represent state and plan space at multiple levels of generality/abstraction (tasks and sub-tasks). While PGPLANNER exposes the state and the current task when querying an expert which can arguably be regarded as some form of explanation, it is far from our objective of robust explanation generation. We generate explanations based on how state literals change via action/control models at varied level of decomposition. We may also expose entire sub-trees (the sequence of all the leaves in the sub-tree results in a partial plan) or we could transfer explanations from other planning tasks.

We can pose the problem of '**learning-to-explain**' as a concurrent sequential decision making problem and use a hierarchical/abstract MDPs to solve it. Such a framework will require us to capture the usefulness of an explanation as the reward signal. For instance, $R^{explain} = \mathcal{F}(R^{env}, R^{human})$ is a combination of the reward signal from the environment and a reward computed via some metric to express user's feedback about usefulness of the explanations. Such a quantity R^{human} may be estimated using #clarifications asked for by the human after the explanation was given. Though we set the problem up in the context of probabilistic hierarchical planning, we intend to investigate generalized solutions that can seamlessly work with imitation learning and predictive modeling.

9.1.3 Deep Understanding of Domains

To realize our objective of intelligent systems being truly robust to density and quality of observations, one of the primary capabilities our AI agent needs, is domain agnosticism. Re-iterating to McCarthy's vision of AI agents having human-like *'common sense'* - we want an agent to leverage human suggestions/advice obtained in some context or domain in other tasks with sparse observations in some other domain(s), similar to how humans use *common sense*. Similarly, cross-domain adaptation of learned higher level concepts is also essential for the agent to fully comprehend advice/knowledge given by the human collaborators. However, if AI agents need extensive engineering and effort from the system designer in order to adapt to different domains, then the purpose and vision of robust agents is defeated.

Domain adaptation and knowledge sharing across different tasks and domains (Higgins et al., 2017; Sener et al., 2016; Wang and Yang, 2011; Gupta et al., 2017; Blitzer et al., 2006; Daume III and Marcu, 2006; Natarajan et al., 2013; Kumaraswamy et al., 2015; Pan et al., 2010) has have studied extensively in the context of sequential decision making (reinforcement learning, planning, imitation learning etc.), statistical learning, probabilistic logic learning, robotics, natural language processing and so on. Most such approaches, though successfully applied to the problem they are designed solve, focus on pair-wise adaptations. That is, given a source domain/task and target domain/task, in some way they estimate some aspect of pairwise distance, be it structural (Kumaraswamy et al., 2015), meta-models (Kang and Feng, 2018), vector space (Higgins et al., 2017; Sener et al., 2016; Wang and Yang, 2011) or some other metric (Dhillon et al., 2012) and then somehow augment the knowledge in the source task such that it fits the syntax and semantics of the target task. The same approach is repeated for any pair of domains encountered. Some leverage the notion of learning with advice/knowledge, human advice is used either to construct a mapping function between the *languages* in source and target, or effectively identify structural patterns that fit the target domain. In all cases discussed so far, a mapping between the source and target languages/encoding is necessary, which either given apriori, obtained as advice, estimated

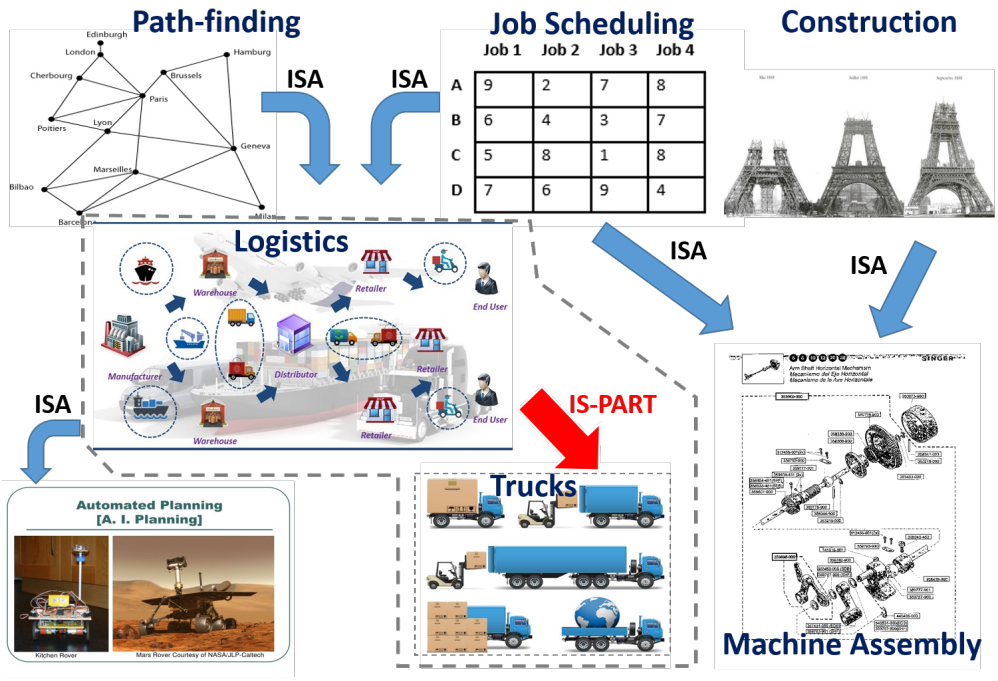


Figure 9.1: Hierarchy/Ontology of domains

via structurally similar components or manifold learning, and that hinders the efficiency of most domain adaptation/transfer approaches which agents cannot afford.

We propose to construct a rich ontology/hierarchy of domains such that the following (no implicit order) becomes naturally achievable without extensive effort or engineering,

- Learning distance metric in a domain space
- Transfer learning in planning/problem-solving, (soft transfer)
- Inducing abstract knowledge from multiple domains
- Domain-based expert selection in multi-expert collaborative learning and decision making
- Cross domain analogical reasoning (Cognitive AI) - akin to *common sense*
- More effective evaluation of learning/decision algorithms

*For instance, in a **fire rescue planning** context the agent can use the concept of 'divert' learned in hospital resource planning.* But if the agent has to spend a lot of time on learning the mapping between two domains and then suggest the 'divert' action in fire rescue, it will be disastrous. Instead, if there is a knowledge ontology/hierarchy and domain adaptation is just a look-up followed efficient reasoning then it achieves the true purpose of a rescue planner. Figure 9.1 shows an example hierarchy among several domains including path planning, job scheduling, logistics, mechanical assembly, rover missions etc. Notice how a domain can be a direct derivative of one or more domains or can be minor modifications of such as well. *Example: logistics is a direct derivative of path finding and job scheduling.*

Domains, when they are designed and encoded for use with AI algorithms have two main aspects, (1) the problem class and (2) the language a domain is encoded in. Our proposed approach will be based on two main ideas,

i. Automated transformation (reduction) to a known or already learned problem class(es) *(Ex: logistics inherits spaces of both to path finding and job scheduling problems; trucks is subsumed by logistics)* (Smith, 1983; Hamfelt et al., 2001; Knoblock, 1991; Creus et al., 2014)

ii. Automated (albeit approximate) mapping between the semantics and syntax of the languages in which different domains are encoded (Russell, 1986; Sarajlić et al., 2016; Shimada et al., 2016; Todorov et al., 2011; Dhillon et al., 2012)

The aim is to investigate the problem from the above mentioned perspectives and learn a progressively growing right knowledge graph of domain concepts (akin to Figure 9.1).

Given: Encoded representation of 2 or more domains $\{d_1, \ldots, d_n\} : n \geq 2$, An initial domain ontology $\mathcal{O}$ (may be empty)

To Do: Perform the following tasks - (1) Determine if the problem space $\rho(d_i)$ can be transformed into the problem space of an existing domain $d_x \sqsubseteq \mathcal{O}$, (2) If there exists such d_x, find a mapping between the languages $\mathcal{L}(d_i), \mathcal{L}(d_x)$, and (3) Update ontology $\mathcal{O}' = \mathcal{O} \cup mapped(d_i)$

9.1.4 Closing Remarks

The frameworks that we have already developed in conjunction with the ones we plan to investigate, are a first step in building out envisioned robust intelligent agent that can learn and act in a complex, structured and noisy world. However, there does exist other aspects beyond the scope of this dissertation that will contribute to making intelligent agents robust to quality and density and we plan to focus on them in future, as we make substantial progress with the objectives outlined here. One key aspect of exploiting *asymmetry of knowledge* (Chapter 1) is the capability of unified knowledge enrichment which includes learning and adding new concepts through dialog and domain space estimation (domain hierarchy/ontology) for which we outline our proposed extensions. Another aspect is bidirectional exchange of ideas. Currently, our HAAI framework (both in sequential decision making as well as knowledge-augmented deep learning) does not validate the preferences, but rather assumes the user is an expert. While the active elicitation framework does pose relevant queries and we know queries communicate ideas to some extent, this dissertation does not investigate richer 'suggestions' that the AI agent may give to the humans. We could extend the framework to recommend improvements to the set of preferences and give other suggestions as well. Also, tracking the expertise of humans to infer advice quality, incorporating other types of advice including feature importance, qualitative constraints, privileged information, etc. as well as investigation of other ideas such as obtaining preferences including crowd of experts, transfer across subtasks and adapting from other domains are potentially impactful future research directions. Finally, our knowledge-augmented deep architecture builds upon CLNs. Extending the idea to other deep models remains an interesting direction for future research. A robust HAAI agent unifying all the ideas presented here will truly be a significant foot forward towards McCarthy's vision of *AI with common sense*.

REFERENCES

Ackley, D. H., G. E. Hinton, and T. J. Sejnowski (1985). A learning algorithm for boltzmann machines. *Cognitive science*.

Ai-Chang, M., J. Bresina, L. Charest, A. Chase, J.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, et al. (2004). Mapgen: Mixed-initiative planning and scheduling for the mars exploration rover mission. *IEEE Intelligent Systems 19*(1), 8–12.

Ali, S. M., N. S. Hadad, and A. M. Jawad (2015). Effect of amoxicillin and cefalexin on the pharmacokinetics of diclofenac sodium in healthy volunteers. *The Medical J Basrah University*.

Allen, J. and G. Ferguson (2002). Human-machine collaborative planning. In *International NASA Workshop on Planning and Scheduling for Space*.

Angles, R. and C. Gutierrez (2008). Survey of graph database models. *ACM Computing Surveys (CSUR)*.

August, J. T., F. Murad, M. Anders, J. T. Coyle, and A. P. Li (1997). *Drug-Drug interactions: scientific and regulatory perspectives*, Volume 43. Academic Press.

Bacchus, F. and F. Kabanza (2000). Using temporal logics to express search control knowledge for planning. *Artificial Intelligence 116*.

Bach, F. R., G. R. Lanckriet, and M. I. Jordan (2004). Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*.

Barber, B. M., T. Odean, and N. Zhu (2009). Systematic noise. *Journal of Financial Markets*.

Bart, E. and S. Ullman (2005). Single-example learning of novel classes using representation by similarity. In *BMVC*.

Battaglia, P., R. Pascanu, M. Lai, D. J. Rezende, et al. (2016). Interaction networks for learning about objects, relations and physics. In *NIPS*.

Becker, M. L. et al. (2007). Hospitalisations and emergency department visits due to drugdrug interactions: a literature review. *Pharmacoepidemiology and Drug Safety 16*.

Berge, C. and E. Minieka (1973). *Graphs and hypergraphs*. North-Holland Publishing Company Amsterdam.

Bhaskara, A., M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan (2010). Detecting high log-densities: An $O(N^{\frac{1}{4}})$ approximation for densest K-subgraph. In *SOTC*.

Blitzer, J., R. McDonald, and F. Pereira (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pp. 120–128. Association for Computational Linguistics.

Blockeel, H. (1999). Top-down induction of first order logical decision trees. *AI Commun. 12*(1-2).

Blum, A. L. and M. L. Furst (1997). Fast planning through planning graph analysis. *Artificial intelligence 90*(1), 281–300.

Boutilier, C., R. Brafman, C. Geib, and D. Poole (1997). A constraint-based approach to preference elicitation and decision making. In *AAAI Spring Symposium on qualitative decision theory*.

Brafman, R. I. and Y. Chernyavsky (2005). Planning with goal preferences and constraints. In *ICAPS*, pp. 182–191.

Braziunas, D. and C. Boutilier (2006). Preference elicitation and generalized additive utility. In *AAAI*.

Bruno, N., R. Ramamurthy, and S. Chaudhuri (2012, May 29). Flexible query hints in a relational database. US Patent 8,190,595.

Bylander, T. (1991). Complexity results for planning. In *IJCAI*.

Callaway, C. B. and J. C. Lester (1997). Dynamically improving explanations: A revision-based approach to explanation generation. In *IJCAI (2)*, pp. 952–958. Citeseer.

Carlson, A., J. Betteridge, B. Kisiel, B. Settles, E. Hruschka Jr, and T. Mitchell (2010). Toward an architecture for never-ending language learning. In *AAAI*.

Castellana, V. G., A. Morari, J. Weaver, A. Tumeo, D. Haglin, O. Villa, and J. Feo (2015). In-memory graph databases for web-scale data. *Computer* (3), 24–35.

Chakraborti, T., S. Sreedharan, and S. Kambhampati (2018). Balancing explicability and explanations. In *AAMAS*.

Chandrasekaran, V., N. Srebro, and P. Harsha (2008). Complexity of inference in graphical models. In *UAI*.

Chen, P. P.-S. (1976). The Entity-Relationship Model–Toward a Unified View of Data. *ACM Transactions on Database Systems (TODS) 1*(1), 9–36.

Chick, H. L. (2007). Teaching and learning by example. *Mathematics: Essential research, essential practice*.

Chung, H. W., B. M. Sadler, and A. O. Hero (2017). Bounds on variance for unimodal distributions. *IEEE Transactions on Information Theory*.

Corby, O., R. Dieng, and C. Hébert (2000). A conceptual graph model for W3C resource description framework. In *ICCS*.

Cortes, C., M. Mohri, and A. Rostamizadeh (2012). Algorithms for learning kernels based on centered alignment. *JMLR*.

Creus, C., P. Fernández, and G. Godoy (2014). Automatic evaluation of reductions between np-complete problems. In *International Conference on Theory and Applications of Satisfiability Testing*.

Csiszár, I. and G. Tusnády (1984). Information Geometry and Alternating minimization procedures. *Statistics and Decisions*.

Darwiche, A. (2003). A differential approach to inference in bayesian networks. *Journal of the ACM*.

Das, M., D. S. Dhami, G. Kunapuli, K. Kersting, and S. Natarajan (2019a). Fast relational probabilistic inference and learning: Approximate counting via hypergraphs. In *AAAI*.

Das, M., D. S. Dhami, G. Kunapuli, K. Kersting, and S. Natarajan (2019b). Fast Relational Probabilistic Inference and Learning: Approximate Counting via Hypergraphs. In *AAAI*.

Das, M., D. S. Dhami, Y. Yu, G. Kunapuli, and S. Natarajan (2019). Knowledge-augmented Column Networks: Guiding Deep Learning with Advice. In *ICML HILL Workshop*.

Das, M., P. Odom, M. R. Islam, J. R. Doppa, D. Roth, and S. Natarajan (2018a). Preference-Guided Planning: An Active Elicitation Approach. In *AAMAS*.

Das, M., P. Odom, M. R. Islam, J. R. J. Doppa, D. Roth, and S. Natarajan (2018b). Planning with actively eliciting preferences. *Knowledge-Based Systems*.

Das, M., Y. Wu, T. Khot, K. Kersting, and S. Natarajan (2016). Scaling Lifted Probabilistic Inference and Learning Via Graph Databases. In *SDM*.

Daume III, H. and D. Marcu (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research 26*, 101–126.

Davis, J. and P. Domingos (2010). Bottom-up learning of markov network structure. In *ICML*.

De Raedt, L., A. Kimmig, and H. Toivonen (2007). Problog: A probabilistic prolog and its application in link discovery. In *AAAI*.

de Salvo Braz, R., E. Amir, and D. Roth (2005). Lifted First Order Probabilistic Inference. In *IJCAI*, pp. 1319–1325.

Demeyer, S., T. Michoel, J. Fostier, P. Audenaert, M. Pickavet, and P. Demeester (2013). The index-based subgraph matching algorithm (ISMA): Fast subgraph enumeration in large networks using optimized search trees. *PLOS One*.

Dhami, D. S., G. Kunapuli, M. Das, D. Page, and S. Natarajan (2018). Drug-drug interaction discovery: Kernel learning from heterogeneous similarities. *Smart Health*.

Dhami, D. S., A. Soni, D. Page, and S. Natarajan (2017). Identifying parkinsons patients: A functional gradient boosting approach. In *AIME*.

Dhillon, P. S., P. P. Talukdar, and K. Crammer (2012). Metric learning for graph-based domain adaptation. In *24th International Conference on Computational Linguistics*, pp. 255.

Diab, A., S. A. Gatz, S. Kapur, D. Ku, C. Kung, P. Hoang, Q. Lu, L. Pogue, Y. K. Shen, N. Shi, et al. (2009, March 3). Search system using search subdomain and hints to subdomains in search query statements and sponsored results on a subdomain-by-subdomain basis. US Patent 7,499,914.

Dietterich, T., A. Ashenfelter, and Y. Bulatov (2004). Training conditional random fields via gradient tree boosting. In *ICML*.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research 13*, 227–303.

Diligenti, M., M. Gori, and C. Sacca (2017). Semantic-based regularization for learning and inference. *AIJ*.

Ding, X., Y. Luo, Q. Li, Y. Cheng, G. Cai, R. Munnoch, D. Xue, Q. Yu, X. Zheng, and B. Wang (2018). Prior knowledge-based deep learning method for indoor object recognition and application. *Systems Science & Control Engineering*.

Domingos, P. and D. Lowd (2009). *Markov Logic: An Interface Layer for AI*. M & C.

Domingos, P. and M. Richardson (2004). Markov logic: A unifying framework for statistical relational learning. In *Proceedings of the SRL Workshop in ICML*.

Dudek, A., A. Frieze, A. Ruciski, and M. ileikis (2013). Approximate counting of regular hypergraphs. *Information Processing Letters*.

Dudek, A., M. Karpinski, A. Ruciński, and E. Szymańska (2014). Approximate counting of matchings in (3,3)-hypergraphs. In *Algorithm Theory – SWAT 2014*.

Džeroski, S., L. De Raedt, and K. Driessens (2001). Relational reinforcement learning. *Machine learning 43*(1-2), 7–52.

Elizalde, F., L. E. Sucar, M. Luque, J. Diez, and A. Reyes (2008). Policy explanation in factored markov decision processes. In *Proceedings of the 4th European Workshop on Probabilistic Graphical Models (PGM 2008)*, pp. 97–104.

Elizalde, F., L. E. Sucar, A. Reyes, and P. deBuen (2007). An MDP approach for explanation generation. In *ExaCt*, pp. 28–33.

Erol, K., J. Hendler, and D. S. Nau (1994). HTN planning: Complexity and expressivity. In *AAAI*.

Fatemi, B., S. M. Kazemi, and D. Poole (2016). A learning algorithm for relational logistic regression: Preliminary results. *arXiv preprint arXiv:1606.08531*.

Feng, F., X. He, Y. Liu, L. Nie, and T.-S. Chua (2018). Learning on partial-order hypergraphs. In *WWW*.

Ferguson, G., J. F. Allen, and B. W. Miller (1996). Trains-95: Towards a mixed-initiative planning assistant. In *AIPS*.

Flach, P. A. (1994). Simply logical intelligent reasoning by example.

Fodor, J. A. (1998). *Concepts: Where cognitive science went wrong*. Oxford University Press.

Fox, M. and D. Long (2003). Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*.

França, M. V. M., G. Zaverucha, and A. S. d'Avila Garcez (2014). Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine Learning*.

Freund, Y., H. Seung, E. Shamir, and N. Tishby (1997). Selective sampling using the query by committee. *Machine Learning 28*, 133–168.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics 29*.

Friend, M., M. Khaled, K. Lefever, and G. Székely (2018). Distances between formal theories.

Fu, L. (1995). Introduction to knowledge-based neural networks. *Knowledge-Based Systems*.

Fung, G., O. Mangasarian, and J. W. Shavlik (2002). Knowledge-based support vector machine classifiers. In *NIPS*.

Fürer, M. and S. P. Kasiviswanathan (2014). Approximately counting embeddings into random graphs. *Combinatorics, Probability and Computing*.

Garcia-Molina, H., J. D. Ullman, and J. Widom (2009). *Database Systems, Second Edition*. Pearson Education, Inc.

Gens, R. and P. Domingos (2013). Learning the structure of sum-product networks. In *ICML*.

Getoor, L., N. Friedman, D. Koller, and A. Pfeffer (2001). Learning probabilistic relational models. *Relational Data Mining, S. Dzeroski and N. Lavrac, Eds.*.

Getoor, L. and B. Taskar (2007). *Introduction to Statistical Relational Learning*. MIT Press.

Ghallab, M., D. Nau, and P. Traverso (2004). *Automated planning: theory & practice*. Elsevier.

Goldberger, J. and E. Ben-Reuven (2017). Training deep neural-networks using a noise adaptation layer. In *ICLR*.

Goldman, R. P. and U. Kuter (2015). Measuring plan diversity: Pathologies in existing approaches and a new plan distance metric. In *AAAI*.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. The MIT Press.

Gove, W. R., J. McCorkel, T. Fain, and M. D. Hughes (1976). Response bias in community surveys of mental health: Systematic bias or random noise? *Social Science & Medicine (1967)*.

Gu, J., P. W. Purdom, J. Franco, and B. W. Wah (1996). Algorithms for the satisfiability (sat) problem: A survey. Technical report, Cincinnati Univ oh Dept of Electrical and Computer Engineering.

Gupta, A., C. Devin, Y. Liu, P. Abbeel, and S. Levine (2017). Learning invariant feature spaces to transfer skills with reinforcement learning. In *ICLR*.

Hamfelt, A., J. F. Nilsson, and N. Oldager (2001). Logic program synthesis as problem reduction using combining forms. *Automated Software Engineering*.

Hamilton, W., Z. Ying, and J. Leskovec (2017). Inductive representation learning on large graphs. In *NIPS*.

Handelman, D., S. H. Lane, and J. J. Gelfand (1990). Integrating neural networks and knowledge-based systems for intelligent robotic control. *IEEE Control Systems Magazine*.

Hayes, A. L., M. Das, P. Odom, and S. Natarajan (2017). User friendly automatic construction of background knowledge: Mode construction from er diagrams. In *KCAP*.

Heckerman, D., C. Meek, and D. Koller (2007). Probabilistic entity-relationship models, PRMs, and plate models. *Introduction to statistical relational learning*, 201–238.

Higgins, I., A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner (2017). Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pp. 1480–1490.

Hoffmann, J. and B. Nebel (2001). The FF planning system: Fast plan generation through heuristic search. *JAIR 14*, 253–302.

Horn, A. (1951). On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*.

Huang, Y.-C., B. Selman, H. Kautz, et al. (1999). Control knowledge in planning: benefits and tradeoffs. In *AAAI/IAAI*, pp. 511–517.

Huynh, T. and R. Mooney (2008). Discriminative structure and parameter learning for Markov logic networks. In *ICML*, pp. 416–423. ACM.

Iordanov, B., K. Vandev, C. Costa, M. Marinov, M. Saraiva de Queiroz, I. Holsman, A. Picard, and I. Bogdahn (2010). HyperGraphDB 1.3.

Jaeger, M. (2007). Parameter learning for relational Bayesian networks. In *ICML*.

Jensen, D. and J. Neville (2002). Linkage and autocorrelation cause feature selection bias in relational learning. In *ICML*.

Jiang, L., Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei (2018). Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*.

Judah, K., A. Fern, P. Tadepalli, and R. Goetschalckx (2014). Imitation learning with demonstrations and shaping rewards. In *AAAI*.

Kang, B. and J. Feng (2018). Transferable meta learning across domains. In *UAI*.

Kaur, N., G. Kunapuli, T. Khot, W. Cohen, K. Kersting, and S. Natarajan (2017). Relational Restricted Boltzmann Machines: A Probabilistic Logic Learning Approach. In *ILP*.

Kautz, H. and B. Selman (1996). Pushing the envelope: Planning, propositional logic, and stochastic search. In *AAAI*.

Kautz, H. A. and B. Selman (1992). Planning as satisfiability. In *ECAI*.

Kazemi, S., D. Buchman, K. Kersting, S. Natarajan, and D. Poole (2014). Relational logistic regression. In *KR*.

Kazemi, S. M. and D. Poole (2018). RelNN: A deep neural model for relational learning. In *AAAI*.

Kersting, K., B. Ahmadi, and S. Natarajan (2009). Counting Belief Propagation. In *UAI*.

Kersting, K. and L. De Raedt (2007). Bayesian logic programming: Theory and tool. In L. Getoor and B. Taskar (Eds.), *An Introduction to Statistical Relational Learning*.

Khan, S. S. and M. G. Madden (2014). One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*.

Khot, T., S. Natarajan, K. Kersting, and J. Shavlik (2011). Learning Markov logic networks via functional gradient boosting. In *ICDM*.

Kimmig, A., S. Bach, M. Broecheler, B. Huang, and L. Getoor (2012). A short introduction to probabilistic soft logic. In *NIPS Workshop*, pp. 1–4.

Kipf, T. N. and M. Welling (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Knoblock, C. A. (1991). Automatically generating abstractions for problem solving. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

Koller, D. (1999). Probabilistic relational models. In *ILP*, pp. 3–13. Springer.

Koller, D. and N. Friedman (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Kozerawski, J. and M. Turk (2018). Clear: Cumulative learning for one-shot one-class image recognition. In *CVPR*.

Krippendorff, K. (1970). Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Kumaraswamy, R., P. Odom, K. Kersting, D. Leake, and S. Natarajan (2015). Transfer learning via relational type matching. In *ICDM*.

Kunapuli, G., K. P. Bennett, R. Maclin, and J. W. Shavlik (2010). The adviceptron: Giving advice to the perceptron. In *ANNIE*.

Kunapuli, G., P. Odom, J. W. Shavlik, and S. Natarajan (2013). Guiding autonomous agents to better behaviors through human advice. In *ICDM*.

Lake, B., R. Salakhutdinov, J. Gross, and J. Tenenbaum (2011). One shot learning of simple visual concepts. In *CSS*.

Lake, B. M., R. Salakhutdinov, and J. B. Tenenbaum (2015). Human-level concept learning through probabilistic program induction. *Science*.

Lanckriet, G. R., N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan (2004). Learning the kernel matrix with semidefinite programming. *JMLR*.

Lang, T. and M. Toussaint (2010). Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research 39*, 1–49.

Lao, N. and W. W. Cohen (2010). Relational retrieval using a combination of path-constrained random walks. *Machine learning 81*(1), 53–67.

Le, Q. V., A. J. Smola, and T. Gärtner (2006). Simpler knowledge-based support vector machines. In *ICML*.

Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*.

Lee, H., P. Pham, Y. Largman, and A. Y. Ng (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*.

Lisi, F. A. (2008). Building rules on top of ontologies for the semantic web with inductive logic programming. *Theory and Practice of Logic Programming*.

Lodhi, H. (2013). Deep relational machines. In *ICONIP*.

Lohman, G. M., E. J. Shekita, D. E. Simmen, and M. S. Urata (2000, July 18). Relational database query optimization to perform query evaluation plan, pruning based on the partition properties. US Patent 6,092,062.

Lowd, D. and P. Domingos (2008). Learning arithmetic circuits. In *UAI*.

Lowd, D. and A. Rooshenas (2013). Learning markov networks with arithmetic circuits. In *AISTATS*.

Lü, L. and T. Zhou (2011). Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*.

Maclin, R. and J. W. Shavlik (1994). Incorporating advice into agents that learn from reinforcements. In *AAAI*.

Maclin, R. and J. W. Shavlik (1996). Creating advice-taking reinforcement learners. *Machine Learning 22*(1), 251–281.

Malec, M., T. Khot, J. Nagy, E. Blask, and S. Natarajan (2016). Inductive logic programming meets relational databases: Efficient learning of Markov logic networks. In *ILP*.

Maloberti, J. and M. Sebag (2004). Fast theta-subsumption with constraint satisfaction algorithms. *Machine Learning*.

McCarthy, J. (1968). Programs with common sense. In *Semantic Information Processing*, pp. 403–418. MIT Press.

Melesko, J. and E. Kurilovas (2018). Semantic technologies in e-learning: Learning analytics and artificial neural networks in personalised learning systems. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*.

Metzler, S. and P. Miettinen (2015). Join size estimation on Boolean tensors of RDF data. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pp. 77–78. International World Wide Web Conferences Steering Committee.

Mihalkova, L. and R. Mooney (2007). Bottom-up learning of Markov logic network structure. In *ICML*, pp. 625–632.

Milch, B., L. S. Zettlemoyer, K. Kersting, M. Haimes, and L. P. Kaelbling (2008). Lifted probabilistic inference with counting formulas. In *AAAI*, Volume 8, pp. 1062–1068.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill Higher Education.

Mitchell, T. M. (1980). *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ. New Jersey.

Miyato, T., S.-i. Maeda, S. Ishii, and M. Koyama (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *TPAMI*.

Molineaux, M. and D. W. Aha (2014). Learning unknown event models. In *AAAI*, pp. 395–401.

Molineaux, M. and D. W. Aha (2015). Continuous explanation generation in a multi-agent domain. Technical report, NAVAL RESEARCH LAB WASHINGTON DC.

Mooney, R. J. (1994). A Preliminary PAC Analysis of Theory Revision. *Computational Learning Theory and Natural Learning Systems*.

Muggleton, S. (1988). Machine invention of first-order predicates by inverting resolution. In *Proc. of 5th Machine Learning Conference*.

Muggleton, S. (1991). Inductive logic programming. *New generation computing*.

Muggleton, S. (1995). Inverse entailment and progol. *New generation computing*.

Muggleton, S. (1996). Stochastic logic programs. In *Advances in Inductive Logic Programming*, pp. 254–264.

Muggleton, S. and L. De Raedt (1994). Inductive logic programming: Theory and methods. *JLP*.

Muggleton, S. and C. Feng (1990). Efficient induction of logic programs. In *ALT*.

Multiple. Drugs.com. (`https://www.drugs.com/drug-interactions/flomax-with-limbitrol-2146-1397-169-8640.html`).

Multiple. drugs.com. (https://www.drugbank.ca/drugs/DB00333).

Multiple. rxlist.com. (https://www.rxlist.com/drug-interactions/glyburide-metformin-oral-and-omeprazole-oral-interaction.htm).

Myers, K. L. (1996). Advisable planning systems. *Advanced Planning Technology*, 206–209.

Narayan-Chen, A., P. Jayannavar, and J. Hockenmaier (2019). Collaborative dialogue in Minecraft. In *ACL*.

Natarajan, S., S. Joshi, P. Tadepalli, K. Kersting, and J. Shavlik (2011). Imitation learning in relational domains: A functional-gradient boosting approach. In *IJCAI*, pp. 1414–1420.

Natarajan, S., K. Kersting, T. Khot, and J. Shavlik (2015). *Boosted Statistical Relational Learners: From Benchmarks to Data-Driven Medicine*. Springer.

Natarajan, S., T. Khot, K. Kersting, B. Gutmann, and J. Shavlik (2012). Gradient-based boosting for statistical relational learning: The Relational Dependency Network case. *MLJ*.

Natarajan, S., T. Khot, K. Kersting, B. Guttmann, and J. Shavlik (2010). Boosting Relational Dependency networks. In *ILP*.

Natarajan, S., G. Kunapuli, C. OReilly, R. Maclin, T. Walker, D. Page, and J. Shavlik (2009). ILP for bootstrapped learning: A layered approach to automating the ILP setup problem. In *ILP*.

Natarajan, S., P. Odom, S. Joshi, T. Khot, K. Kersting, and P. Tadepalli (2013). Accelerating imitation learning in relational domains via transfer by initialization. In *ILP*.

Natarajan, S., A. Prabhakar, N. Ramanan, A. Baglione, K. Connelly, and K. Siek (2017). Boosting for postpartum depression prediction. In *CHASE*.

Natarajan, S. and P. Tadepalli (2005). Dynamic Preferences in Multi-Criteria Reinforcement Learning. In *ICML*.

Natarajan, S., P. Tadepalli, T. G. Dietterich, and A. Fern (2009). Learning first-order probabilistic models with combining rules. *AMAI*.

Nau, D. S., T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman (2003). Shop2: An htn planning system. *J. Artif. Intell. Res.(JAIR) 20*, 379–404.

Neumann, T. and G. Moerkotte (2011). Characteristic sets: Accurate cardinality estimation for rdf queries with multiple joins. In *ICDE*.

Neville, J. and D. Jensen (2007). Relational dependency networks. In L. Getoor and B. Taskar (Eds.), *Introduction to Statistical Relational Learning*, pp. 653–692. MIT Press.

Niepert, M. (2012). Lifted probabilistic inference: An mcmc perspective. In *Statistical Relational AI Workshop at UAI*, Volume 2012.

Nies, A. T., U. Hofmann, C. Resch, E. Schaeffeler, M. Rius, and M. Schwab (2011). Proton pump inhibitors inhibit metformin uptake by organic cation transporters (octs). *PLoS One*.

Nigam, S. K. (2015). What do drug transporters really do? *Nature reviews. Drug discovery 14*(1), 29.

Niu, F., C. Ré, A. Doan, and J. W. Shavlik (2011). Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS. *PVLDB*.

Niyogi, P. and F. Girosi (1996). On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. *Neural Computation*.

Noordewier, M. O., G. G. Towell, and J. W. Shavlik (1991). Training knowledge-based neural networks to recognize genes in dna sequences. In *Advances in neural information processing systems*.

Odom, P., T. Khot, R. Porter, and S. Natarajan (2015). Knowledge-based probabilistic logic learning. In *AAAI*.

Odom, P. and S. Natarajan (2016a). Active advice seeking for inverse reinforcement learning. In *AAMAS*.

Odom, P. and S. Natarajan (2016b). Actively interacting with experts: A probabilistic logic approach. In *ECML*.

Odom, P. and S. Natarajan (2018). Human-guided learning for probabilistic logic models. *Frontiers in Robotics and AI journal special issue on Statistical Relational Artificial Intelligence*.

Pan, J. Z. (2009). Resource description framework. In *Handbook on Ontologies*, pp. 71–90. Springer.

Pan, S. J., Q. Yang, et al. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*.

Parr, R. and S. J. Russell (1998). Reinforcement learning with hierarchies of machines. In *NIPS*, pp. 1043–1049.

Patrini, G., A. Rozza, A. K. Menon, R. Nock, and L. Qu (2017). Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*.

Pavicić, M., A. Van Winkelhoff, and J. De Graaff (1991). Synergistic effects between amoxicillin, metronidazole, and the hydroxymetabolite of metronidazole against actinobacillus actinomycetemcomitans. *Antimicrobial agents and chemotherapy*.

Pazzani, M. J. (1992). An information-based approach to integrating empirical and explanation-based learning. *ILP*.

Pham, T., T. Tran, D. Q. Phung, and S. Venkatesh (2017). Column networks for collective classification. In *AAAI*.

Poole, D. (2003). First-Order Probabilistic Inference. In *IJCAI*.

Poon, H. and P. Domingos (2007). Joint inference in information extraction. In *AAAI*, pp. 913–918.

Poon, H. and P. Domingos (2011). Sum-product networks: A new deep architecture. In *UAI*.

Poon, H., P. Domingos, and M. Sumner (2008). A general method for reducing the complexity of relational inference and its application to mcmc. In *AAAI*, pp. 1075–1080.

Poyrekar, S., S. Natarajan, and K. Kersting (2014). A deeper empirical analysis of cbp algorithm: Grounding is the bottleneck. In *International Workshop on Statistical Relational AI (StarAI)*.

PrudHommeaux, E., A. Seaborne, et al. (2008). Sparql query language for rdf. *W3C recommendation 15*.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine learning*.

Raedt, L. D., K. Kersting, S. Natarajan, and D. Poole (2016). Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lect. on AI & ML*.

Rahman, T., P. Kothalkar, and V. Gogate (2014). Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *ECMLPKDD*.

Ramanan, N., M. Das, K. Kersting, and S. Natarajan (2019). Discriminative non-parametric learning of arithmetic circuits. In *Working Notes of the ICML Workshop on Tractable Probabilistic Models (TPM)*.

Ramanan, N., G. Kunapuli, T. Khot, B. Fatemi, S. M. Kazemi, D. Poole, K. Kersting, and S. Natarajan (2018). Structure learning for relational logistic regression:an ensemble approach. In *KR*.

Ratliff, N., D. Silver, and A. Bagnell (2009). Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 25–53.

Ratner, A. J., C. M. De Sa, S. Wu, D. Selsam, and C. Ré (2016). Data programming: Creating large training sets, quickly. In *NIPS*.

Ravkic, I., M. Znidarsic, J. Ramon, and J. Davis (2018). Graph sampling with applications to estimating the number of pattern embeddings and the parameters of a statistical relational model. *Data Min. Knowl. Discov. 32*.

Richards, B. L. and R. J. Mooney (1992). *Learning relations by pathfinding*. Artificial Intelligence Laboratory, University of Texas at Austin.

Richardson, M. and P. Domingos (2006). Markov logic networks. *Machine learning 62*(1-2), 107–136.

Ricks, B., B. Thuraisingham, and P. Tague (2018). Lifting the smokescreen: Detecting underlying anomalies during a ddos attack. In *ISI*.

Riondato, M., M. Akdere, U. Çetintemel, S. B. Zdonik, and E. Upfal (2011). The vc-dimension of sql queries and selectivity estimation through sampling. In *Machine Learning and Knowledge Discovery in Databases*, pp. 661–676. Springer.

Rocktäschel, T., M. Bošnjak, S. Singh, and S. Riedel (2014). Low-dimensional embeddings of logic. In *ACL 2014 Workshop on Semantic Parsing*.

Rooshenas, A. and D. Lowd (2014). Learning sum-product networks with direct and indirect variable interactions. In *ICML*.

Rooshenas, A. and D. Lowd (2016). Discriminative structure learning of arithmetic circuits. In *AISTATS*.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan.

Ross, S. and D. Bagnell (2010). Efficient reductions for imitation learning. In *AISTATS*.

Rouveirol, C. (1990). Saturation: Postponing choices when inverting resolution. In *ECAI*.

Rouveirol, C. (1992). Extensions of inversion of resolution applied to theory completion. *ILP*.

Russell, S. (1986). *Analogical and Inductive Reasoning*. Ph. D. thesis, Stanford University.

Sallans, B. and G. E. Hinton (2004). Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*.

Sammut, C. and R. B. Banerji (1986). Learning concepts by asking questions. *Machine learning: An artificial intelligence approach*.

Sarajlić, A., N. Malod-Dognin, Ö. N. Yaveroğlu, and N. Pržulj (2016). Graphlet-based characterization of directed networks. *Scientific reports 6*, 35098.

Sarkhel, S., D. Venugopal, T. Pham, P. Singla, and V. Gogate (2016). Scalable training of Markov logic networks using approximate counting. In *AAAI*.

Sarne, D. and B. J. Grosz (2007). Estimating information value in collaborative multi-agent planning systems. In *AAMAS*.

Schapire, R. and Y. Freund (2012). *Boosting: Foundations and Algorithms*. The MIT Press.

Schiefer, B., L. G. Strain, and W. P. Yan (1998, June 2). Method for estimating cardinalities for query processing in a relational database management system. US Patent 5,761,653.

Sener, O., H. O. Song, A. Saxena, and S. Savarese (2016). Learning transferrable representations for unsupervised domain adaptation. In *NIPS*.

Seputis, E. A. (2000, January 4). Database system with methods for performing cost-based estimates using spline histograms. US Patent 6,012,054.

Settles, B. (2010). Active learning literature survey. Technical report, University of Wisconsin.

Shavlik, J. and S. Natarajan (2009). Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *IJCAI*.

Shavlik, J. W. and G. G. Towell (1989). Combining explanation-based learning and artificial neural networks. In *Proceedings of the sixth international workshop on Machine learning*. Elsevier.

Shimada, Y., Y. Hirata, T. Ikeguchi, and K. Aihara (2016). Graph distance for complex networks. *Scientific reports 6*, 34944.

Shortliffe, E. H. (1974a). Mycin: a rule-based computer program for advising physicians regarding antimicrobial therapy selection. Technical report.

Shortliffe, E. H. (1974b). A rule-based computer program for advising physicians regarding antimicrobial therapy selection. In *Proceedings of the 1974 annual ACM conference-Volume 2*, pp. 739–739. ACM.

Silberschatz, A., H. F. Korth, S. Sudarshan, et al. (1997). *Database system concepts*.

Singla, P. and P. Domingos (2008). Lifted first-order belief propagation. In *AAAI*, pp. 1094–1099.

Slota, G. M. and K. Madduri (2013). Fast approximate subgraph counting and enumeration. In *Intl. Conf. on Parallel Processing*.

Smith, D. R. (1983). A problem reduction approach to program synthesis. In *IJCAI*, pp. 32–36.

Snowden, L. R. (2003). Bias in mental health assessment and intervention: theory and evidence. *American Journal of Public Health*.

Sohrabi, S., J. A. Baier, and S. A. McIlraith (2009). HTN planning with preferences. In *IJCAI*.

Sohrabi, S. and S. A. McIlraith (2008). On planning with preferences in htn. In *(NMR)*.

Srinivasan, A. (2004). *The Aleph Manual*.

Srinivasan, A. (2007). Aleph: A learning engine for proposing hypotheses. *Software available at http://web2. comlab. ox. ac. uk/oucl/research/areas/machlearn/Aleph/aleph. pl*.

Srinivasan, A., R. D. King, S. H. Muggleton, and M. Sternberg (1997). Carcinogenesis predictions using ILP. *Inductive Logic Programming*.

Stocker, M., A. Seaborne, A. Bernstein, C. Kiefer, and D. Reynolds (2008). Sparql basic graph pattern optimization using selectivity estimation. In *WWW*.

Stroulia, E. and A. K. Goel (1994). Learning problem-solving concepts by reflecting on problem solving. In *ECML*.

Sun, Z., H. Wang, H. Wang, B. Shao, and J. Li (2012). Efficient subgraph matching on billion node graphs. *Proceedings of the VLDB Endowment 5*(9), 788–799.

Sutskever, I., J. B. Tenenbaum, and R. R. Salakhutdinov (2009). Modelling relational data using bayesian clustered tensor factorization. In *NIPS*.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., D. Precup, and S. Singh (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*.

Tadepalli, P., R. Givan, and K. Driessens (2004). Relational reinforcement learning: An overview. In *Proceedings of the ICML-2004 workshop on relational reinforcement learning*, pp. 1–9.

Talamadupula, K., G. Briggs, M. Scheutz, and S. Kambhampati (2013). Architectural mechanisms for handling human instructions in open-world mixed-initiative team tasks. *Advances in Cognitive Systems (ACS) 6*.

Tan, S.-W. and J. Pearl (1994). Specification and evaluation of preferences for planning under uncertainty. In *KR*.

Taskar, B., P. Abbeel, M. Wong, and D. Koller (2007). Relational Markov networks. *Introduction to statistical relational learning*, 175–200.

Taskar, B., M.-F. Wong, P. Abbeel, and D. Koller (2004). Link prediction in relational data. In *NIPS*.

Tax, D. M. J. (2001). *One-class classification: Concept learning in the absence of counter-examples.* Ph. D. thesis, TU Delft.

Thoolen, M., B. Wilfert, A. Jonge, P. Timmermans, P. Zwieten, et al. (1984). Effect of salbutamol and the pde-inhibitor ra 642 on the clonidine withdrawal syndrome in rats. *Auton Autacoid Pharmacol*.

Todorov, K., P. Geibel, and C. Hudelot (2011). A framework for a fuzzy matching between multiple domain ontologies. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pp. 538–547. Springer.

Toh, K. C., M. J. Todd, and R. H. Tütüncü (1999). SDPT3 – A Matlab software package for semidefinite programming, v. 1.3. *OMS*.

Torrey, L., T. Walker, J. Shavlik, and R. Maclin (2005). Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *ECML*, pp. 412–424.

Towell, G. and J. Shavlik (1994). Knowledge-based artificial neural networks. *Artificial intelligence 70*(1-2), 119–165.

Vadhan, S. P. (2001). The complexity of counting in sparse, regular, and planar graphs. *SICOMP*.

Valiant, L. G. (1979). The complexity of enumeration and reliability problems. *SICOMP*.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*.

Venugopal, D. and V. Gogate (2014). Evidence-based clustering for scalable inference in markov logic. In *ECML-PKDD*.

Venugopal, D., S. Sarkhel, and V. Gogate (2015). Just count the satisfied groundings: Scalable local-search and sampling based inference in MLNs. In *AAAI*.

Vilar, S., E. Uriarte, L. Santana, T. Lorberbaum, G. Hripcsak, C. Friedman, and N. P. Tatonetti (2014). Similarity-based modeling in large-scale prediction of drug-drug interactions. *Nature protocols*.

Vinyals, O., C. Blundell, T. Lillicrap, D. Wierstra, et al. (2016). Matching networks for one shot learning. In *NIPS*.

Vitányi, P. M. (2013). Conditional kolmogorov complexity and universal probability. *Theoretical Computer Science*.

Vitányi, P. M., F. J. Balbach, R. L. Cilibrasi, and M. Li (2009). Normalized information distance. In *Information theory and statistical learning*.

Šourek, G., V. Aschenbrenner, F. Železny, and O. Kuželka (2015). Lifted relational neural networks. In *NIPS Workshop on Cognitive Comput.: Integr. Neural & Symbolic Approaches*.

Walker, T., G. Kunapuli, N. Larsen, D. Page, and J. Shavlik (2011). Integrating knowledge capture and supervised learning through a human-computer interface. In *KCAP*.

Wang, H.-Y. and Q. Yang (2011). Transfer learning by structural analogy. In *AAAI*.

Wang, Y.-X., R. Girshick, M. Hebert, and B. Hariharan (2018). Low-shot learning from imaginary data. In *CVPR*.

Wilson, R. H. and J.-C. Latombe (1994). Geometric reasoning about mechanical assembly. *Artificial Intelligence*.

Xu, Y., A. Fern, and S. Yoon (2009). Learning linear ranking functions for beam search with application to planning. *JMLR*, 1571–1610.

Xu, Y., A. Fern, and S. W. Yoon (2010). Iterative learning of weighted rule sets for greedy search. In *ICAPS*.

Yang, S., M. Korayem, K. AlJadda, T. Grainger, and S. Natarajan (2017). Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach. *Knowledge-Based Systems*.

Yang, S. and S. Natarajan (2013). Knowledge intensive learning: Combining qualitative constraints with causal independence for parameter learning in probabilistic models. In *ECML-PKDD*.

Yoon, S., A. Fern, and R. Givan (2008). Learning control knowledge for forward search planning. *JMLR 9*(Apr), 683–718.

Yoon, S. W., A. Fern, and R. Givan (2007). Ff-replan: A baseline for probabilistic planning. In *ICAPS*.

Yoon, S. W., A. Fern, R. Givan, and S. Kambhampati (2008). Probabilistic planning via determinization in hindsight. In *AAAI*.

Zeng, Q., J. M. Patel, and D. Page (2014). Quickfoil: scalable inductive logic programming. *Proceedings of the VLDB Endowment*.

Zhao, H., P. Poupart, and G. J. Gordon (2016). A unified approach for learning the parameters of sum-product networks. In *NIPS*.

BIOGRAPHICAL SKETCH

Mayukh Das is a PhD candidate in the Department of Computer Science at The University of Texas at Dallas, advised by Professor Sriraam Natarajan. His research interests include human-AI collaboration, sequential decision making, reinforcement learning, automated planning, statistical relational models and approximation. He completed his Master of Science (MS) degree in Computer Science from Indiana University, Bloomington. He obtained his Bachelor of Technology (B. Tech) in Computer Science & Engineering from West Bengal University of Technology (Kolkata, WB, India).

Before pursuing graduate studies, Mayukh used to work as a software developer for several years with some of the leading information technology (IT) service companies in India, such as Tata Consultancy Services Ltd. and Ericsson. During his career with the software industry his expertise was in developing analytics and large-scale data warehouse and business intelligence (BI) systems.

CURRICULUM VITAE

Mayukh Das

October 15, 2016

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd., ECSS 4.613
Richardson, TX 75080-3021, U.S.A.

Email: mayukh.das1@utdallas.edu

Educational History:

B.Tech., Computer Science & Engineering, West Bengal University of Technology, 2009
M.S., Computer Science, Indiana University, Bloomington, 20015
PhD, Computer Science, The University of Texas at Dallas, 2019

Human-Allied Efficient and Effective Learning in Noisy Domains
PhD Dissertation
Computer Science Department, The University of Texas at Dallas
Advisors: Dr. Sriraam Natarajan

Employment History:

Research Assistant, The University of Texas at Dallas, August 2017 – present
Research Assistant, Indiana University Bloomington, May 2015 – August 2017
Teaching Assistant, Indiana University Bloomington, August 2013 – August 2015
Data Modeling Intern (Summer Intern), Openwords LLC, May 2014 – August 2014
Solution Integrator, Ericsson India Global Services Limited, Jan 2012 – April 2013
Assistant System Engineer, Tata Consultancy Services Ltd., Dec 2009 – Dec 2011

Professional Services:

Program Committee: AAAI 2020, SDM 2020, CODS-CoMAD 2020
Journal Reviewer: IEEE TPAMI, Elsevier KBS, 2019
Assistant Electronic Publishing Editor: JAIR 2018 – present

Publications:

Journal Papers:
1. M. Das, P. Odom, Md. R. Islam, J.R. Doppa, D. Roth and S. Natarajan, "Planning with Actively Eliciting Preferences," in *Knowledge-Based Systems (Elsevier),* 2018 .

Conference Papers:

2. M. Das, N. Ramanan, and S. Natarajan, "One-shot Concept Learning with Human Advice: An Iterative Rule Learning Approach," *under review*.
3. M. Das, Y. Yu, D. Dhami, G. Kunapuli and S. Natarajan, "Human-Guided Learning of Column Networks: Augmenting Deep Learning with Advice," under review at *NeurIPS '19*.
4. M. Das, D. Dhami, G. Kunapuli, K. Kersting, and S. Natarajan, "Fast Relational Probabilistic Inference and Learning: Approximate Counting via Hypergraphs," in *AAAI*, 2019.
5. M. Das, D. Dhami, G. Kunapuli, K. Kersting, and S. Natarajan, "Approximate Counting for Fast Inference and Learning in Probabilistic Programming," in *PROBPROG*, 2018. `[Short; Poster]`.
6. D. Dhami, G. Kunapuli, M. Das and S. Natarajan, "Drug-Drug Interaction Discovery: Kernel Learning from Heterogeneous Similarities," in *IEEE/ACM CHASE*, 2018.
7. M. Das, P. Odom, Md. R. Islam, J.R. Doppa, D. Roth and S. Natarajan, "Preference-Guided Planning: An Active Elicitation Approach," in *AAMAS*, 2018.
8. A. Hayes, M. Das, P. Odom and S. Natarajan, "User Friendly Automatic Construction of Background Knowledge: Mode Construction from ER Diagrams," in *K-CAP*, 2017.
9. M. Das, Y. Wu, T. Khot, K. Kersting and S. Natarajan, "Scaling Lifted Probabilistic Inference and Learning via Graph Databases," in *SDM*, 2016.

Workshop Papers:
10. M. Das, D. Dhami, Y. Yu, G. Kunapuli and S. Natarajan, "Knowledge-augmented Column Networks: Guiding Deep Learning with Advice," in *HILL workshop @ ICML '19*.
11. N. Ramanan, M. Das, K. Kersting and S. Natarajan, "Discriminative Non-Parametric Learning of Arithmetic Circuits," in *TPM workshop @ ICML '19*.
12. A. Narayan-Chen, C. Graber, M. Das, Md. R. Islam, S. Dan, S. Natarajan, J. R. Doppa, J. Hockenmaier, M. Palmer, D. Roth, "Towards Problem Solving Agents that Communicate and Learn," in *First Workshop on Language Grounding for Robotics @ ACL*, 2017.
13. M. Das, Md. R. Islam, J.R. Doppa, D. Roth and S. Natarajan, "Active Preference Elicitation for Planning," in *HMCL workshop @ AAAI*, 2017.
14. M. Das, Y. Wu, T. Khot, K. Kersting and S. Natarajan, "Graph Based Approximate Counting for Relational Probablistic Models," in *StaRAI workshop @ UAI* 2015.