THE UNIVERSITY
of EDINBURGH

Learning to Make Decisions with Unforeseen Possibilities

Craig Innes

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2019

Abstract

Methods for learning optimal policies often assume that the way the domain is *conceptualised*—the possible states and relevant actions that are needed to solve one's decision problem—is known in advance and does not change during learning. This is an unrealistic assumption in many scenarios. Often, new evidence can reveal important information about what is *possible*, not just what is likely, or unlikely. A learner may have been completely unaware such possibilities even existed prior to learning.

This thesis presents a model of an agent which discovers and exploits unforeseen possibilities from two sources of evidence: domain exploration and communication with an expert. The model combines probabilistic and symbolic reasoning to estimate *all* components of the decision problem, including the set of belief variables, the possible actions, and the probabilistic dependencies between variables. Unlike prior work on solving decision problems by discovering and learning to exploit unforeseen possibilities (e.g., Rong (2016); McCallum and Ballard (1996)), our model supports discovering and learning to exploit unforeseen *factors*, as opposed to an additional atomic state. Becoming aware of an unforeseen factor presents computational challenges when compared with becoming aware of an additional atomic state, because even a boolean factor doubles the size of the decision problem's hypothesis space as opposed to increasing it by just one more state. We show via experiments that one can meet those challenges by adopting (defeasible) reasoning principles that are familiar from the literature on belief revision: roughly, default to simple models over more complex ones and default to conserving what you've learned from prior evidence.

For one-step decision problems, our agent learns the components of a Decision Network; for sequential problems, it learns a Factored Markov Decision Process. We prove convergence theorems for our models, given the learner's and expert's strategies for gathering evidence. Furthermore, our experiments show that the agent converges on optimal behaviour even when it starts out completely unaware of factors that are critical to success.

Lay Summary

When learning to how to identify the best plans of action in a given environment, humans have the remarkable ability to adapt what they already know to totally new and unforeseen information. For example, when we are taught a new skill, we can easily imagine how we might apply that skill in scenarios we have encountered previously. Similarly, when we ask someone a question about what will happen in a given situation, their answer might not just merely tell us what is most likely from a list of outcomes we had already anticipated. Instead, they might inform us of an outcome that we had never even thought of. Despite not expecting this answer, we can integrate such information into our previous understanding of the world without much difficulty.

Most artificial intelligence systems, on the other hand, have become very good at analysing large amounts of data so as to find the optimal solution to a given task, but only provided that all the actions the system is allowed to take, as well as all the outcomes that can possibly occur are explicitly specified in advance. If, mid-way through the task, the system is informed of some previously unforeseen action it can take, or encounters an unforeseen concept that it so far hasn't considered when distinguishing one state of affairs from another, then the system will not know how to adapt what it's already learned to incorporate these unforeseen possibilities and then learn to exploit them.

This thesis takes a different approach and presents an artificial intelligence system that learns not just from raw data, but also from advice from an informed expert. In this way, our system mirrors the teacher-apprentice model of learning, which is common between humans. Our informed expert can interject during the task with advice and corrections, which may include concepts our system was previously completely unaware of. We provide a method by which our system can integrate this new unforeseen concept into its existing model of a task, and thereby continually grow its awareness of the given problem.

Acknowledgements

First, I would like to thank my supervisor Alex Lascarides for her continuous support of my PhD study and related research, for their encouragement, feedback, and academic knowledge, and for their guidance in navigating the confusing world of university bureaucracy.

I would also like to thank Subramanian Ramamoorthy, Benjamin Rosman, Stefano Albrecht, Vaishak Belle, and other members of faculty who have taken the time to look at various iterations of my research and provided their own perspective and valuable criticisms.

I thank my fellow students in the Institute for Language, Cognition and Computation, as well as in the Centre for Doctoral Training for Data Science. Sharing the stresses of research life over four years of lunches on the second floor has been vitally cathartic.

Lastly, thanks to my family and friends for supporting me, and reminding me of the world outside of writing this thesis.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Craig Innes*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

This thesis explores the problem of an agent learning optimal policies when it starts unaware of factors that are critical to solving its task. In typical machine learning tasks, a learner starts with some fixed set of hypotheses, but is uncertain about which of them is the most likely (and is perhaps also uncertain about the rewards they will obtain from the possible outcomes of their actions). In this thesis, "unawareness" has a much stronger meaning, concerning the ignorance of something's *existence* as opposed to its relative likelihood—what if the learner was initially unaware of the set of hypotheses itself?

Consider the following medical example: Suppose a decision making agent prescribes a drug which it has learned from past experience is effective at treating a certain illness. However, a senior pharmacologist later objects to the prescription based on a reason totally unforeseen by the agent—the patient carries a newly discovered genetic trait, and the drug produces harmful side effects in its carriers. Instead, the pharmacologist suggests prescribing to the patient a new topical cream (which the agent had not previously heard of), which is slightly less effective at treating the illness, but does not produce harmful side effects. These unforeseen discoveries (becoming aware of the existence of the new genetic trait and new topical cream) may occur after the agent has already learned quite a lot about how other (foreseen) factors impact the drug's effectiveness.

This example illustrates at least three challenges, the combination of which typically is

not handled by current methods for learning optimal behaviour (Chapter 2 contains a detailed discussion of related work):

1. In addition to starting ignorant of the probabilistic structure of the decision problem, the relative likelihood among possible states and the range of rewards they engender, the agent starts unaware of even the *true set of possible actions and states themselves*.

2. Domain exploration alone might not be enough to discover these unknown factors. For instance, it is unlikely the agent would discover the concept of a new genetic trait by just continuing to prescribe drugs and observing their effect. *Expert instruction* may be necessary to overcome unawareness.

3. An expert might interject with relevant advice *during* learning, not just at the beginning when the initial model of the problem is designed.

As Coenen *et al.* (2017) point out, such scenarios are common in human discussion—the answer to a question may not only provide information about which of the questioner's existing hypotheses are likely, but also reveal unforeseen consequences not yet considered. This example also shows that it is often infeasible for an agent to gather an exhaustive explanation of all relevant factors *prior to* learning. A domain expert may find it difficult or impossible to express all details of a decision problem in advance due to lack of knowledge, cognitive constraints on either the expert or learner, or limitations on communication. However, it may be easy for an expert to offer contextually relevant advice *during* learning.

There are many other examples of such scenarios are in the real world. The invention of vaccinations or the creation of new financial instruments are examples of discovering new actions in medicine and finance. The discovery of syphilis added a previously unknown consequence to the act of sex.[1] Another example is in robotic skill learning. Methods like Cakmak and Thomaz (2012) enable an expert to teach a robot *how* to perform a new action, but not necessarily *when* it is optimal to use it (the latter task being the focus of this thesis). In lifelong learning scenarios, there is a need for flexible and robust responses to a vast array of contingencies (Silver, 2011; Thrun and Pratt, 2012). We would like such a robot to continually add to its knowledge, and revise the hypothesis space of possible states and actions within which decisions are made. In this context, there is a need for autonomous model management (Liebman *et al.*,

[1] These examples are from Karni and Viero (2013)

2017), which calls for reasoning about what the hypothesis space is, in addition to policy learning within those hypothesis spaces.

Current learning and decision making models do not address such issues; they assume the task is to use data to *refine* a distribution over a *fixed* hypothesis space. In these frameworks, any change to the set of possible hypotheses constitutes a unrelated problem. The above examples, however, illustrate a sort of *reverse bayesianism* (Karni and Viero, 2013), where the hypothesis space itself expands over time.

An inevitable consequence of a learner starting out unaware of critical factors is that its conceptualisation of the domain might be deficient: it fails to discriminate among states where a crucial factor (which the agent is unaware of) takes different values. In light of this, one might be tempted to side-step the difficult problem of *overcoming* unawareness of states and actions, and instead represent unawareness *implicitly* via an infinite number of hidden variables (Doshi-Velez, 2009; Wood *et al.*, 2006), or abandon learning a structured model and use densely connected layers of hidden nodes to learn optimal behaviour directly from raw sensory input (Mnih *et al.*, 2015). Deep reinforcement learning in particular (e.g. Mnih *et al.* (2015)) has proved useful for learning implicit representations of large problems, particularly in domains where sensory input is complex and high-dimensional (e.g. computer vision). However, in many such models, the focus of attention for abstraction and representation learning is on perceptual features rather than probabilistic relations (see e.g. (Pearl, 2018)) and other decision-centric attributes. For instance, in work by Chen *et al.* (2015), who demonstrate an end-to-end solution to the problem of autonomous driving, the decisions are not elaborated beyond "follow the lane" or "change lanes" although significant perceptual representation learning may need to happen in order to work with the predicate "lane" within the raw sensory streams (often relying on millions or billions of data trials). For safety critical reasons, or ones involving significant investment (e.g. driving a car, advising on a medical procedure, deciding on crops to grow for the year) it is important that a system can explain the reasoning behind its decisions so the user can trust its judgement. This issue lies behind recent major funding initiatives in *explainable AI* (e.g., DARPA-BAA-16-53 (2016)).

We instead propose a system which explicitly overcomes its unawareness, and constructs an interpretable structured model of the problem.

1.2 Contributions

This thesis presents an agent that, in a complementary approach to representation learning in batch mode from large data sets, uses evidence and a reasoning mechanism to *incrementally* construct an *interpretable* model of a decision problem from which optimal policies can be computed. The main contributions of this thesis are:

1. We provide a model for learning in which an agent starts unaware of the existence of factors on which an optimal policy depends, then learns optimal behaviour via direct experience and expert advice. The learning proceeds *incrementally* such that, at every step of the task, the agent is able to produce a compact and *interpretable* model representing its beliefs about the variables and probabilistic structure of the task. Crucially, the agent can revise *all* components of its model. This means not just the probabilistic dependencies between variables (Chapter 3), but also the domain of the reward function, and the set of state variables and actions itself (Chapters 4 and 5).

2. We specify a communication framework via which an expert offers advice to the agent *during* learning: that is, advice is offered in reaction to the agent's latest attempts to solve the task, rather than all expert input being provided prior to learning. By offering *contextual* advice in a piecemeal manner, we have worked towards ensuring the interaction between the agent and expert builds upon dialogue moves that would be quite natural between a *human* expert and learner (for example, correcting the agent when they make a sub-optimal move, or helping the agent resolve a conflict between its current model and the latest piece of evidence). This contrasts with fully exhaustive explanations of a task, which may be impossible for a human expert to articulate due to lack of knowledge, cognitive constraints, or limitations on the method of communication.

   Under our model, the expert advice can include mentions of factors which the agent didn't know existed, thereby exposing the agent to unforeseen possibilities. This advice provides important *qualitative* information to complement the *quantitative* information conveyed by statistical correlations in the domain trials.

   What is not covered in this thesis is the pervasive ambiguity that is inherent in natural language, which is beyond the scope of our work. Rather, the signals that the agent and expert exchange are more like formal semantic representations

of natural language utterances, once parsed and disambiguated as the speaker intended (see Sections 4.2.1 and 5.3.1 for details).

3. We state theorems and proofs which show that our agent-expert dialogue strategy is one where the learner is guaranteed to eventually become aware of all the actions and variables necessary to express the true optimal policy (see Theorems 1 and 4 on pages 59 and 93).

   Our model also guarantees that the learner, at all times, assumes a (graphical) model of its decision problem which is well-formed and consistent with all evidence it has observed so far (see Theorem 2, page 60). We can therefore guarantee that the learner's model can be used to compute what the learner currently believes to be an optimal policy. This is typically not an issue for standard learning tasks where the hypothesis space is completely known in advance of learning, but is one which must be carefully considered when evidence can expand a learner's conceptualisation of the decision problem.

4. We conduct experiments across a suite of both hand-crafted problems from existing literature (which have been chosen on the grounds that they represent practical real-world problems) and randomly generated decision problems (which demonstrate our method's robustness to problems of varying complexity). These experiments demonstrate that, in practice as well as in theory, the agent can learn the optimal policy from evidence, even when initially unaware of variables that are critical to its success (Sections 4.4, 5.4)

1.3 Task Overview and Scope

For a given decision problem, our agent's overarching goal is to learn a model which guides it towards an *optimal decision policy*. As is traditional in AI, we adopt a Savagean model of rational action (Savage, 1972): the agent's action is an optimal trade-off between what it prefers and what it believes it can achieve. More formally, for a one-step decision problem, this means the agent will take the action a^* which, given observed evidence e, maximizes its *expected utility*:

$$a^* = \arg\max_{a \in A} EU(a|e) = \arg\max_{a \in A} \sum_{s \in \mathcal{S}} P(s|a,e)\mathcal{R}(s) \tag{1.1}$$

Here, $\mathcal{R}(s)$ is the reward function, which gives the real-valued reward received for entering state s. The distribution $P(s|a,e)$ gives the probability of entering state s given observed evidence e and that the agent took action a.

As is standard in learning tasks, the agent may not initially know $\mathcal{R}$ or $P(s|a,e)$, and so must estimate them via trial-and-error in the task, or by receiving advice from an informed expert. However, the core issue in this thesis is that the agent faces the extra difficulty of starting unaware of certain possible actions $a \in A$, or is unaware of certain states $s \in \mathcal{S}$ on which the optimal policy depends. To put this another way, the agent starts out not knowing the set A nor the state space $\mathcal{S}$. The task then, is to use evidence from both domain exploration and expert advice to learn the optimal policy, despite beginning with an initial model defined over an incomplete set of possible states and actions, with incorrect dependencies among them, and a possibly incorrect reward function. Each chapter addresses increasingly complex versions of this core scenario.

Chapter 3 sets up some preliminaries for the rest of the thesis by describing a method for learning the probabilistic structure of a problem from data. Our key contribution here is to show that existing methods for learning *Bayesian Networks* (BNs) can be extended to the case where not all belief variables are known prior to learning, but are added *incrementally* during learning. Chapter 4 then builds on this initial framework to show how an agent can learn not just the probabilistic structure of a problem, but also learn an optimal policy for taking actions within single-stage environments. This chapter also shows the mechanism by which the agent actually overcomes its initial unawareness. Chapter 5 then extends the agent's capabilities further to deal with *sequential* tasks, where the agent's current actions may affect not just its current immediate reward, but also have long-term effects on future states and rewards.

To restrict the scope of our task, this thesis deals exclusively with decision problems where all variables have **finite** and **discrete** values. The tasks considered are also all **stationary**. This means that the true properties of the underlying decision problem do not change over time. Further we assume that all variables are **fully observable** once an agent becomes aware of them. This means that, for example, if an agent becomes aware of a variable X_i at time step t, then the agent can observe the value of this variable at all subsequent time steps $t' > t$.[2]

[2]We do not, however, make the much stronger assumption that the agent can re-perceive past domain trials upon discovering an unforeseen factor. In our example, this means that the agent cannot go back and observe the value of X_i at time steps $t' < t$.

The primary goal of the work in this thesis is to show that, under our model, it is possible for an agent to overcome its initial unawareness and learn all relevant factors necessary to learn the true optimal policy for a given task. Additionally however, we also investigate whether the intuitive principles used to guide the design of our learner actually result in better performance.

One of these principles is that of **Minimality**. Minimality is essentially a form of Occam's razor, which means that we prefer simple models over complex ones. This preference already underpins most existing model-based learning techniques (Koller, 2009), as simpler models not only make inference and learning more computationally tractable, but also tend to generalize better to unseen data.

We also investigate the value of **Conservativity** in incremental learning. Conservativity is the compelling intuition that when the learner discovers a new unforeseen possibility, they should preserve as much as possible about what they inferred from past evidence, even when this new discovery forces them to revise their current model to remain consistent with new evidence. Conservativity underlies existing symbolic models of common-sense reasoning (Poole, 1993; Hobbs *et al.*, 1993; Alchourrn *et al.*, 1985), the acquisition of probabilistic dependencies (Bramley *et al.*, 2016; Buntine, 1991) and preference change (Hansson, 1995; Cadilhac *et al.*, 2015). These principles are deliberately stated in broad terms here in the introduction, as they influence design decisions across many parts of the learning model. We will highlight throughout the thesis the explicit algorithms, formulas and properties of the model where these principles manifest.

In general, the results of this thesis offer several key insights:

- It is feasible to build agents that learn optimal policies even when they start out unaware of the existence of almost all critical factors, and even on relatively large decision problems (e.g., those featuring upwards of a million atomic states).
- Using algorithms which favour minimal complexity allows one to keep the learning of complex decision problems tractable without harming the learner's ability to eventually converge upon the optimal policy (Section 4.4).
- Having an agent conserve its previous beliefs upon discovering previously unforeseen factors has a number of advantages. Not only does it prevent the quality of the agent's learned model from degrading throughout learning (Section 3.3), but also results in faster convergence to the optimal policy in terms of both

computational time and number of trials (Sections 4.4 and 5.4).

- Depending on the application domain, one can choose to trade-off the quality of the agent's learned policy against the amount of time the expert spends giving advice to the agent. This can be done by altering the expert's *tolerance* towards the agents mistakes (Sections 4.4 and 5.4).

Chapter 2

Related Work

This chapter provides an overview of related literature dealing with both symbolic and statistical approaches to decision-making and learning. In particular, it highlights how such works deal with *unawareness*, and the gap our work fills in this area.

2.1 Decision Theory and Preference

The traditional analysis of rational action optimises the trade-off between what you prefer and what you believe can achieve (Savage, 1972). This notion is typically formalized as maximizing *expected utility* (See equation (1.1) in the introduction). However, such formalisms assume the hypothesis space — the set of possible states $\mathcal{S}$ and actions $\mathcal{A}$—is fixed and known in advance. Changing the hypothesis space under this framework essentially amounts to considering an unrelated decision problem. In this thesis, our agent discovers new states and actions through growing awareness, but instead of treating the expanded hypothesis space as an unrelated problem, it uses its experience before the discovery to inform its behaviour and inferences after the discovery.

Some recent works do integrate unawareness into a theory of decision making. Karni and Viero (2013) present a set of representation theorems and update principles which formalize an agent's growing awareness about a hypothesis space. Such representations are framed as *Reverse Bayesianism*: Bayesian methods consider the complete hypothesis space, then *shrink* the space of plausible states as evidence reveals which states are unlikely. Reverse Bayesian methods start with an incomplete view of the hypothesis

space and *expand* the set of plausible states as evidence reveals new possibilities. Such work is valuable in that it shows that a principled theory of rational action exists even in the face of expanding awareness, and many of the theorems for updating beliefs which result as a consequence of the axioms of their representation (e.g. that the likelihood ratios over existing beliefs should be preserved when new possibilities are introduced) align with the intuitive principle of Conservativity (Section 1.3) that the model in this thesis is based upon.

This thesis builds on such theorems to explore how an agent could *discover* these gaps in awareness in the first place via trial and error and dialogue (4.2.1). Additionally, Karni and Viero (2013) remain neutral on the specific *structure* the agent's probabilistic beliefs should take, and consequently provide no algorithms which support modelling dependencies among probabilistic factors within a given domain. Such probabilistic structure allows one to exploit probabilistic independencies during learning and inference, and are therefore necessary to scale these tasks to cope with the complexity of practical and realistic decision problems (e.g., it is not unusual for the domain to admit millions of atomic states). This thesis explores how such updates on growing awareness might be applied to explicit structured representations, such as decision networks and factored markov decision processes. These allow us to analyse larger problems by providing a compact representation of the joint probability distribution of the problem.

One further difference is that, while Karni and Viero (2013) provide a set of important foundational theorems, they do not test the performance of their models empirically. This thesis focuses on evaluating the empirical performance of such approaches to unawareness in a concrete implementation of a learning agent.

Karni and Viero (2017) go further, allowing the agent to imagine that their current conception of the hypothesis space is incomplete as part of its decision policy, even if the agent is unsure *what* they might be unaware of. Agents who do not consider this possibility are "Myopic", since they always evaluate actions as if their current view of the hypothesis space is complete, even if they have expanded their awareness in the past. These considerations present an interesting problem, but are outside the scope of this thesis. The agent presented in subsequent sections considers the utility of actions in the "Myopic" sense described above.

Discovering a new state or action may prompt revisions of the reward function itself rather than just the probability distribution over belief states. However, current models

of preference which support revision on discovering an unforeseen possibility assume a qualitative preference model (Hansson, 1995; Cadilhac *et al.*, 2015). Following Cadilhac *et al.* (2015), we use evidence to dynamically construct an ever more specific partial description of preferences (with the agent defaulting to indifference to produce a complete preference function from the partial description). Unlike Cadilhac *et al.* (2015), the learning agent uses evidence to estimate *numeric* payoffs rather than qualitative preference relations.

2.2 Unawareness Logics and Game Theory

There is a growing body of work on how to model unawareness as an epistemic concept within agents (Fagin and Halpern, 1987; Board *et al.*, 2011). These theories deal with the problem of developing a logically precise model of the agent's unawareness of particular facts.

Other work approaches these issues from a game-theoretic perspective (Feinberg, 2004; Halpern and Rego, 2013; Heifetz *et al.*, 2013) and focus on when the agent's unawareness causes it to deviate from truly optimal behaviour (and how an adversary might *exploit* such gaps in awareness).

However, all such work interprets awareness from an omniscient perspective which is capable of modelling all possible options. A fully aware agent (e.g. the domain expert in this thesis) or an analyst can model an unaware agent in this way, but it does not characterise the unaware agent's *own subjective perspective* (Li, 2008). In contrast, our learning task requires the unaware agent to use evidence to expand and change its set of possible options.

2.3 Structure Learning for Decision Problems

As part of its task, the agent must learn and revise an interpretable model of the decision problem to incorporate newly discovered actions and concepts. Hence, part of the task is to incrementally learn the domain's probabilistic structure. Several approaches exist for jointly learning probabilistic dependencies in an online setting (Bramley *et al.*, 2016; Buntine, 1991; Friedman, 1998; Alcobe, 2005) and also for exploiting causal structure

to speed-up inference (Albrecht and Ramamoorthy, 2016).

This thesis extends this work in three main ways to meet the objectives described in Chapter 1. First, current models for learning dependencies all assume that the vocabulary of belief variables does not change during learning, but in our task it changes when the agent becomes aware of an unforeseen action or concept. Second, since optimal actions depend on payoffs as well as beliefs, this thesis integrates learning dependencies with learning potential pay-offs. Third, our model uses both domain trails and expert advice as evidence. Methods such as Buntine (1991) can technically support the incorporation of expert evidence too, but only if the messages are about probabilistic relations and are all declared *before* learning. In contrast, our model *interleaves* dialogue and learning. This enables the expert to offer advice in a timely and contextually relevant manner, which means they can explain particular outcomes and correct mistakes. There are many scenarios where a human expert cannot articulate to full extent their skill and expertise in a lengthy monologue prior to learning; they can, however, react in a competent way to specific scenarios that the learner and expert encounter as the learner attempts to learn the task.

There has been some work on inferring the probabilistic structure and utilities of other agents using *influence diagrams* (Bielza *et al.*, 2010; Nielsen and Jensen, 2004; Suryadi and Gmytrasiewicz, 1999). More broadly, work on *inverse reinforcement learning* (Rafferty *et al.*, 2015; Herman *et al.*, 2016; Wu *et al.*, 2018) attempts to estimate an agent's beliefs and preferences based on observing their actions. However, the majority of such work assumes the task is simply to learn a reward function for the agent given full knowledge of its probabilistic beliefs about the world, or attempts to estimate probabilistic parameters given a fixed, known structure. Our work considers the scenario in which the agent may start unaware of the probabilistic structure, or even the set of variables which are relevant to optimal behaviour.

One alternative way to think about modelling probabilistic dependencies in an incomplete hypothesis space is to represent factors the agent may be unaware of as a potentially infinite number of latent (hidden) variables. Such *non-parametric* approaches to structure learning (e.g. Wood *et al.* (2006)) leverage Monte-Carlo sampling techniques to form a distribution over a number of potential structures with an unbounded number of latent variables. This technique is valuable for discovering latent structure within an existing data set. However, such methods stop short of seeking out explanations for what these latent structures actually represent in terms of the underlying decision problem.

Our model instead takes the approach of attempting expand and repair the model when such deficiencies are discovered, linking unforeseen possibilities to explicitly named state and action variables via contextual expert advice.

2.4 Reinforcement Learning and Sequential Decisions

The literature on learning optimal policies for sequential decision problems includes methods for dealing with numerous types of change, such as noise in sensing and actuation (Da Silva *et al.*, 2006), and non-stationary rewards (Besbes *et al.*, 2014). Such methods have also been applied in situations involving hidden variables (e.g. with Partially Observed Markov Decision Processes (Leonetti *et al.*, 2011)) or in cases where the learner must reason about the underlying probabilistic structure of the problem (e.g. with Factored Markov Decision Processes (Degris *et al.*, 2006)). However, these methods require all possible states and actions to be known in advance of learning. In contrast, our agent learns these components from evidence.

There are a few notable exceptions in the reinforcement learning (RL) literature, which attempt to relax the standard assumption that all possible states and actions are known in advance of learning. Rong (2016) define Markov Decision Processes with Unawareness (MDPUs), which can learn optimal behaviour when an agent starts unaware of some actions. They apply MDPUs to a robotic-motion task with around 1000 discretised atomic states. The agent uses an *explore* move, which randomly reveals useful motions they were previously unaware of.

Our work differs from theirs in several ways. First, we provide a concrete mechanism for discovering unforeseen factors via expert advice, rather than random discovery from the agent's own exploration. This gets us closer to supporting interactive task learning (Laird *et al.*, 2017) of the kind which happens between a human apprentice and expert. Second, we allow the agent to discover *explicit belief variables* rather than atomic states, and focus more on exploiting the inherent structure present in problems with large numbers of features. This enables us to conduct experiments in Section 4.4 with upwards of a million distinct atomic states (as compared to the roughly 1000 atomic states in the experiments of Rong (2016)).

Another exception is the family of *selective perception* methods, such as the U-tree algorithm of McCallum and Ballard (1996). These methods were designed for problems

where the true state space may technically be available, but is infeasibly large to reason with. The U-tree method initially treats all states as indistinguishable, then recursively splits states in two as the agent discovers significant differences in their expected reward. The agent thus learns an increasingly complex representation of the state space as and when observable evidence justifies the increase in complexity. The U-Tree algorithm is a powerful approach to handling large sequential decision problems. However, there are three main learning tasks that our model supports that the U-tree algorithm does not. First, our model accommodates situations where the agent is made aware of an entirely new possible action, while the U-Tree algorithm assumes that the set of actions remains fixed. Secondly, our model learns from (qualitative) evidence provided by a domain expert, as well as evidence from trial and error. Thirdly, our model learns unforeseen *concepts* and probabilistic relations among them, while the U-tree algorithm does not reason about which dimensions define the state space, nor their probabilistic dependencies. This third difference is related to the second: we ultimately want our model to enable an agent to learn about the domain and how to behave in it from a human expert, and it is quite natural for a human to justify why one state is difference from another on the basis of concepts on which optimal behaviour depends by saying, for instance "wrapping an item in bubble-wrap makes it less likely to break". Our model supports learning from this type of evidence.

As explained in Chapter 5, model-based sequential problems require us to provide not just a graphical structure that captures the probabilistic dependencies among the variables defining the hypothesis space, but also to one to exploit the structure present in the reward and policy functions. To maintain tractability throughout the problem, our method uses decision trees (Boutilier *et al.*, 2000), but there is a wide literature on alternative structured representations of rewards and policies (Hoey *et al.*, 1999; Guestrin *et al.*, 2003; Boutilier *et al.*, 2001; van Otterlo, 2009). The focus of this thesis however is not on exploring the most efficient structured representations (something which is often domain-specific) but rather on showing generally how such structured representations can be built upon to accommodate a hypothesis space which grows with the agent's own awareness.

Deep reinforcement learning (Mnih *et al.*, 2015) also aims to learn a suitable abstraction of the data to enhance convergence toward optimal policies. Deep reinforcement learning combines deep learning (i.e. a convolutional network or CNN (Bengio, 2013; Hinton *et al.*, 2006)) with reinforcement learning, and has proven extremely useful (e.g.

see Chouard (2016)). However, the major successes in deep reinforcement learning have not so far considered changing state-action spaces in the sense of the examples discussed in Chapter 1 (e.g., discovering a new genetic trait or possible treatment in the case of the medical example). Instead, the focus has been on using a large number (typically, in the millions or tens of millions) of domain trials to re-describe a large state-action space in terms of abstracted versions that render policy learning more tractable.

As argued previously however, we believe that without evidence conveyed by an outside expert, an agent may never find out the parts that are missing from their initial model. If we wish to accommodate a learning process which allows the integration of expert advice during learning, then our model for learning must be incremental and be able to handle updates from qualitative information as well as quantitative information. Unfortunately, expert evidence about causal relations ("X causes Y") or preference ("If C is true, then doing X is better than doing Y") are inherently symbolic. Adjusting weights in a sub-symbolic representation like a CNN so that it satisfies certain qualitative properties is currently an open problem. Another limitation is that implicit models generated by DRL lack explainability–they are unable to elaborate on *why* a given action was recommended over another. Our approach complements methods like DRL by dynamically building an *interpretable model* of the decision problem. Such interpretable models make it easier for the agent to explain the reasons behind a given decision, and also allow the agent to evaluate its interpretation of expert messages against its current conceptualisation of the decision problem via the standard logical technique of model checking.

2.5 Dialogue and Active Learning

There are several areas of research in which expert evidence is used to improve the performance of a learning agent. In *Transfer Learning*—where knowledge of the optimal policy for a *source task* is used to improve performance on a related *target task* (Taylor and Stone, 2009)—the transfer of knowledge is sometimes captured via an expert "giving advice" to a learner (Torrey *et al.*, 2006). Typically, both the source and target task must belong to the same domain. If they do not, an explicit mapping is usually provided from the states and actions in the source task to the states and actions in the target task. In contrast, our agent incrementally learns a *single* task, but is

occasionally made aware of new concepts, which it must learn to accommodate into its current knowledge, which in turn has been learned from prior evidence. Another difference is that in transfer learning, the expert advice is all declared before the agent begins learning. In our model, the agent and expert engage in a dialogue throughout learning.

Knox and Stone (2009, 2010) use human expert evidence to inform RL, but they confine this evidence to updating the likely outcomes of actions and their rewards; they do not support cases where expert evidence reveals information that the agent *was not aware was possible*. On the other hand, researchers have developed models for learning optimal policies via a combination of domain actions and natural language instructions, where the instructions may include neologisms, whose semantics the agent must learn (e.g. Liang (2005)). However, this work assumes that the neologism denotes an already known concept within the agent's abstract planning language. For example, on encountering the neologism "block", the agent's task is to learn that it maps to the concept `block` that is already an explicit part of the agent's domain model. In contrast, we support learning optimal policies when the neologism denotes a concept that the agent is currently *unaware* of; e.g. to support learning when `block` is not part of the agent's conceptualisation of the domain.

Forbes *et al.* (2015) use embodied natural language instructions to support teaching a robot a new skill (a task known as *learning by demonstration*). Their model learns how to map a natural language neologism to what might be a novel combination of sensory values—in this sense, the meaning of the neologism may be an unforeseen concept—and this novel concept then informs the task of learning new motor controls. In effect, this work links a rapidly growing body of research on learning how to ground natural language neologisms in embodied environments (known as the *symbol grounding problem* (Siskind, 1996; Hristov *et al.*, 2017) with learning a new skill. What it does not support is *integrating* the newly acquired skill into an existing hypothesis space consisting of other actions, consequences and rewards; so they do not support learning optimal policies when both this new skill *and* other actions (and related concepts) are needed.

Our aim in this thesis is to supply a complimentary set of learning algorithms to this prior work. Like Forbes *et al.* (2015), our agent learns from both trial and error and from instruction. Instead of focussing, as they do, on learning *how* to execute a new skill however, we instead focus on learning *when* it is optimal to execute it. Given

that we wish to support larger planning tasks, we also broaden the goals to discovering and learning to exploit arbitrary unforeseen concepts, not just the learning of spatial concepts that Forbes *et al.* (2015) are limited to (see also Dobnik *et al.* (2012)). On the other hand, natural language ambiguity makes extracting the hidden message from natural language utterances a highly complex process (Grice, 1975; Bos *et al.*, 2004; Zettlemoyer and Collins, 2007; Reddy *et al.*, 2016). We bypass this complexity by using a *formal* language as the medium of conversation. This formal language can be broadly construed as the kind of language one uses to represent the output of natural language semantic parsing (Artzi and Zettlemoyer, 2013). Replacing the expert that we deploy in our experiments with an expert *human* would require linking the model we present in this thesis with existing work on grounded natural language acquisition and understanding. Additionally, we also bypass learning how to compute referents that are denoted by the speaker's message in the visual environment (i.e. we do not tackle the problem of grounded language acquisition). Instead, when the speaker utters a neologism, we simply supply the agent with the ability to sense its value in subsequent states that the agent experiences. Ultimately, to build an agent which learns directly from natural language input from a human teacher, one would need to combine our current model (which models how to exploit unforeseen domain factors) with natural language acquisition and understanding techniques. This is an ambitious task which is beyond the scope of this thesis, but forms a major focus of our future work.

In the taxonomy of traditional machine learning tasks, interaction with an expert (or *oracle*) is often categorised as an instance of *active learning* (Settles, 2009). In active learning, rather than passively observing evidence from a sequence of domain trials, a learning agent is given some control over which kinds of evidence it observes. Often this control is exercised by requesting evidence from an expert.

Murphy (2001) describes an active learning framework in which the agent can fix the value of certain state variables in each upcoming observation. This effectively allows the agent to perform causal interventions, allowing them to more effectively discover the causal dependencies between variables. Masegosa and Moral (2013) take this further and allow the agent to ask about explicit edges in the probabilistic structure itself. In both these papers however, they assume that the set of state variables is fixed and known in advance. This means, for example, that the expert cannot tell the agent that a probabilistic dependency exists between a variable that the agent asked about and some entirely new variable which the agent is currently unaware of. As mentioned previously,

our model allows for expert evidence which may result in an expansion of the agent's current vocabulary. Another minor difference is that, to evaluate which question about probabilistic dependency will offer the largest information gain, the agent in Masegosa and Moral (2013) must calculate a probability distribution over all possible probabilistic structures (or approximate one using monte-carlo sampling methods) using the entire batch of data observed so far. Such a procedure is expensive and difficult to integrate into an on-line learning scenario. In contrast, our agent asks questions in response to conflicts between the current piece of evidence and its current model, or when it explicitly fails to learn a valid model.

Lakkaraju *et al.* (2017) tackle learning "unknown unknowns" via interaction with an oracle. This work addresses a specific type of unawareness: an agent assigns an incorrect label to a trial with high confidence. Crucially however, they assume that the *correct* label for this trial must be a label that the agent is *already currently aware of*. In other words, they exclude the option that the hypothesis space itself is incomplete, which is the type of unawareness that we are interested in.

Chapter 3

Incremental Learning of Bayesian Networks with Expanding Awareness

In this chapter, we introduce the concept of *unawareness* as it relates to probabilistic beliefs. We start by showing how an agent incrementally learns both the probabilistic structure and parameters of a problem by learning a *Bayesian Network*. We then show how an agent can adapt what it has learned so far upon becoming aware of previously unforeseen variables.

The contribution of this chapter is to take existing incremental methods for learning BNs and adapt them to account for the discovery of unforeseen variables during learning. We show through experiments on several well-known BN learning problems that our learner converges to an accurate model of the underlying distribution (and accurate representation of that distribution's structure) even when initially unaware of important belief variables. Further, by conserving information learned from past data each time a new belief variable is discovered, our model is able to learn with more consistency than one which abandons such information to start learning "from scratch" in the newly expanded hypothesis space.

In later chapters, we build on these initial techniques by applying them to *decision problems* where an agent must deal with unawareness of *actions* and *rewards*, as well as just belief variables. Further, we will later build on the abstract notion of "expanding one's awareness" outlined in this chapter by providing a concrete mechanism by which an agent can become aware of unforeseen factors via discussion with an informed expert.

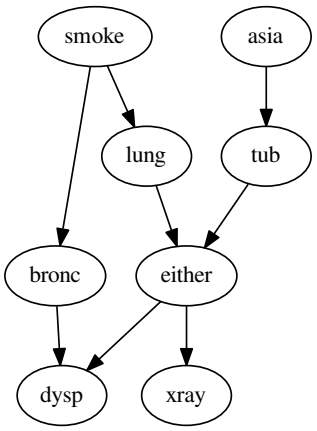


Figure 3.1: The *Asia* BN. Arrows between nodes indicate probabilistic dependencies between variables

We begin by summarizing bayesian networks and techniques to learn them when the agent is fully aware of all belief variables in the hypothesis space (but is ignorant of the conditional independence among variables in the domain, and of the probability distributions). In subsequent chapters, we then extend this problem to one where the agent begins unaware of relevant variables in the hypothesis space.

3.1 Learning Bayesian Networks with Full Awareness

Bayesian networks (BNs) decompose the state space of a problem into a set of belief variables $\mathcal{X} = \{X_1, \ldots X_n\}$. Each state s is a joint assignment to all variables in $\mathcal{X}$ (i.e., $s \in v(\mathcal{X})$). A BN is a directed acyclic graph (DAG) denoted Pa, and parameters θ which exploit the conditional independencies between variables in $\mathcal{X}$. For each $X \in \mathcal{X}$, $Pa_X \in Pa$ is the *parent set* of X. Given Pa_X, the value of X is conditionally independent of all $Y \in \mathcal{X} \setminus Pa_X$ which are non-descendants of X. The parameters $\theta_X \in \theta$ then define the conditional probability distributions $\theta_X = P(X|Pa_X)$. Figure 3.1 gives an example of the *Asia* network used in some of the experiments at the end of the chapter. We can see for example that a patient's chances of getting tuberculosis are independent of whether they smoke, have lung cancer, etc., if we know whether they have been to Asia (because $Pa_{tub} = \{asia\}$).

Given $\mathcal{X}$, we can learn the most likely structure Pa^* and parameters θ^* from observed data $D_{0:t} = [d_0, \ldots, d_t]$ (where $d_i \in v(\mathcal{X})$) by learning $P(Pa|D_{0:t})$ and $P(\theta|D_{0:t}, Pa)$. A common way to calculate $P(Pa|D_{0:t})$ is to use the *Bayesian Dirichlet Score* (BD-Score) (Heckerman *et al.*, 1995):

$$P(Pa|D_{0:t}) \propto \prod_{X \in \mathcal{X}} \mathrm{BD}_t(X, Pa_X) \tag{3.1}$$

$$\mathrm{BD}_t(X, Pa_X) = P(Pa_X) \prod_{j \in v(Pa_X)} \frac{\beta(N^t_{0|j} + \alpha_{0|j}, \ldots, N^t_{m|j} + \alpha_{m|j})}{\beta(\alpha_{0|j}, \ldots, \alpha_{m|j})} \tag{3.2}$$

Here, $\beta(n_1, \ldots, n_m)$ is the *multivariate beta function*, and $N^t_{X=i|Pa_X=j}$ is the number of states in $D_{0:t}$ where X has assignment i and its parents have joint assignment j.[1] For example, the term $N^t_{lung=1|smoke=0}$ describes the number of trials in $D_{0:t}$ where the patient has lung cancer, but doesn't smoke. The $\alpha_{i|j}$ parameters come from the *dirichlet prior* over θ and act as a "pseudo-count" when data is sparse:

$$Dir(\theta_{X|j}|\vec{\alpha}) = \frac{\prod_{i \in v(X)} \theta^{\alpha_{i|j}}_{i|j}}{\beta(\alpha, \ldots, \alpha)} \tag{3.3}$$

We typically choose the prior $P(Pa)$ to penalize complex structures, as in equation (3.4) which assigns a cost of $\rho < 0.5$ for each extra parent:

$$P(Pa_X) = \rho^{|Pa_X|}(1-\rho)^{|\mathcal{X} \setminus Pa_X|} \tag{3.4}$$

3.1.1 Reducing the space of "Reasonable" Parent Sets

Unfortunately, evaluating the full posterior of (3.1), or even finding its maximum, is infeasible for even a moderate number of variables, as the number of possible BNs is hyper-exponential on the size of $\mathcal{X}$ (Tian and He, 2009). To tackle this, most methods approximate Pa^* via either local search (Teyssier and Koller, 2005), sampling methods (Madigan *et al.*, 1995), or by restricting the search space of structures to those considered

[1] Where the context is clear, we have compressed this notation to $N^t_{i|j}$

reasonable, given evidence so far (Buntine, 1991). For our task, we have another issue—most BN learning methods (with some notable exceptions (Buntine, 1991; Friedman and Goldszmidt, 1997; Alcobe, 2005; Yasin and Leray, 2013)) operate *once* on a single *batch* of data, and thus consider collecting the *sufficient statistics* (i.e., the BD-scores of each Pa_X and their associated $N_{i|j}$) to be a negligible pre-computed cost. In a decision problem however, agents gather data *incrementally* and exploit their current beliefs during learning, so we must *update* the sufficient statistics at each time step.

To address these issues, we follow Buntine (1991) by constructing a *reasonable parent lattice* $\mathcal{P}_X$ for each X. Starting from $\emptyset$, we construct larger parent sets by combining sets seen so far, and track the highest scoring set according to (3.2). Any parent-set with a score lower than some proportion κ of the best score is considered "unreasonable", and is not expanded further.[2] This makes Buntine's model defeasibly minimal, in the sense that it assumes that adding further variables to an unreasonable parent set won't ever make it reasonable.

Figure 3.2 shows an example lattice for the possible parent sets of X_1. Starting from the empty set, the algorithm first computes the BD-scores of all size 1 parent sets—$\{X_2\}, \{X_3\}, \{X_4\}$, and $\{X_5\}$. The BD-score of X_5 falls below the threshold proportion κ of the highest scoring parent set X_2, so it is marked as "unreasonable" and not considered further. Next, the algorithm considers all size 2 combinations of the remaining reasonable parent sets — $\{X_2\}, \{X_3\}$, and $\{X_4\}$. At this point, $\{X_3, X_4\}$ falls below the minimum threshold, and so is also marked as "unreasonable". This process continues until either the lattice is fully expanded, or all remaining leaf nodes are marked unreasonable.

3.1.2 Finding the Optimal Structure with ILP

Once we have our reduced set $\mathcal{P}_X$ for each X, we can find the combination of reasonable parent sets which maximize (3.1) and form a valid DAG. In our task, we will eventually be learning $\mathcal{X}$, so cannot assume we know a total-ordering $\prec$ over $\mathcal{X}$ (as Buntine (1991) does), and thus cannot choose each Pa_X^* independently. For instance, it might be that $\{asia\}$ is locally the most likely parent set for tub, and also that $\{tub\}$ is the most likely

[2] In contrast to Buntine (1991), since we do not know $\prec$ in advance, we also consider all *subsets* of a reasonable parent set to be reasonable. This makes it so it is always possible to construct at least one valid DAG, regardless of how aggressively the search space is pruned.

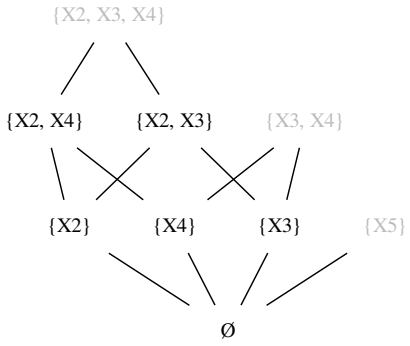


Figure 3.2: Parent lattice for X_1. Grey nodes are unreasonable

parent set for *asia*, but we cannot include both $Pa_{tub} = \{asia\}$ and $Pa_{asia} = \{tub\}$ in our final BN without creating a cycle.

Instead, following Bartlett and Cussens (2017), we treat finding Pa^* as the linear program given below:

$$\begin{aligned} \max \quad & \sum_{X \in \mathcal{X}} \sum_{Pa_X \in \mathcal{P}_X} I(Pa_X \to X) \ln[BD(X, Pa_X)] \\ \text{s.t.} \quad & \sum_{Pa_X \in \mathcal{P}_X} I(Pa_X \to X) = 1 \qquad \forall X \\ & \sum_{\substack{X \in C \\ Pa_X \cap C = \emptyset}} I(Pa_X \to X) \geq 1 \qquad \forall C \subseteq \mathcal{X} \\ & I(Pa_X \to X) \in \{0,1\} \quad \forall X, Pa_X \end{aligned} \tag{3.5}$$

The variables $I(Pa_X \to X)$ denote whether we chose Pa_X as X's parent set, while the constraints ensure each X has only one parent set and that the parent sets form a DAG.

Once we have learned the most likely structure Pa^*, we can learn its most likely parameters θ^* via (3.6):

$$\mathbb{E}_{\theta | D_{0:t}, \Pi}(\theta_{i|j}) = \frac{N_{i|j} + \alpha_{i|j}}{N_{\cdot|j} + \alpha_{\cdot|j}} \tag{3.6}$$

Our method has three main advantages. First, reducing the number of reasonable parent sets and using the modular *BD*-score means we can *incrementally* update the probability of each Pa_X using past calculations of (3.2) via (3.7):[3]

[3]We include a proof of (3.7) in the appendix.

$$\mathrm{BD}_t(X, Pa_X) = \mathrm{BD}_{t-1}(X, Pa_X) \frac{N^t_{i|j} + \alpha_{i|j} - 1}{N^t_{\cdot|j} + \alpha_{\cdot|j} - 1} \tag{3.7}$$

Second, expressing structure learning as a linear program makes it easy to add *extra* structural constraints later (as we do in Chapter 4). Third, each lattice $\mathcal{P}_X$ implicitly captures an *approximate distribution* over each X's parents.[4] The lattice in figure 3.2 for instance approximates the posterior distribution $P(Pa_{X_1}|D_{0:t})$. In Section 3.2, the agent uses this distribution to conserve information as its awareness of $\mathcal{X}$ expands.

Algorithm 1 gives an overview of how the agent updates its current beliefs about Pa^* and θ^* given the most recent trial d_t. In most time steps, the agent simply incrementally updates its previous sufficient statistics for each of the reasonable parent sets, then chooses the combination of parent sets which maximizes the linear program given by (3.5). However, if a significant amount of time τ has passed, we may wish to update the set $\mathcal{P}$ of reasonable parent sets by re-running the lattice update method described in Section 3.1.1

Algorithm 1 Updating BN (Fixed Belief Variables)

function UPDBN($\mathcal{P}^{t-1}, N^{t-1}_{i|j}, d_t$)
  if $t \equiv 0 \pmod{\tau}$ **then**
    Revise $\mathcal{P}^t, N^t_{i,j}$, $P(Pa_X|D_{0:t})$ via lattice update (Buntine, 1991) (threshold κ)
  else
    $\mathcal{P}^{t-1} \leftarrow \mathcal{P}^t$
    $N^t_{i,j} \leftarrow$ Update with $d_t, N^{t-1}_{i,j}$ for all $i \in v(X), j \in v(Pa_X), X \in \mathcal{X}, Pa_X \in \mathcal{P}^t_X$
    $P(Pa_X|D_{0:t}) \leftarrow$ Calculate via (3.7) for all $Pa_X \in \mathcal{P}^t_X, X \in \mathcal{X}$
  $Pa^* \leftarrow$ Maximize (3.5)
  $\theta^* \leftarrow$ Calculate via (3.6)
  return $\langle N^t_{i,j}, \mathcal{P}^t, Pa^*, \theta^* \rangle$

3.2 Adapting to growing awareness

The previous sections assumed the learner was aware (in advance of learning) of the full set of belief variables $\mathcal{X}$ which define the hypothesis space of possible BNs. In this

[4] Technically, without $\prec$, each $\mathcal{P}_X$ is a distribution over *markov blankets* for X, but we enforce acyclicity, so this is not an issue.

Time	X_1	X_2
0	1	0
⋮	⋮	⋮
$t-1$	1	0

Time	X_1	X_2	**Z**
0	1	0	**?**
⋮	⋮	⋮	⋮
$t-1$	1	0	**?**
t	0	0	**1**

Table 3.1: View of data before and after discovering Z.

section, we drop this requirement.

As an example, suppose the learner is initially aware of only two variables: $\mathcal{X}^0 = \{X_1, X_2\}$, and $\mathcal{X}^0 \subset \mathcal{X}^+$, where $\mathcal{X}^+$ is the true set of variables. We assume the learner cannot observe the value of variables it is unaware of. In the medicine example from the introduction for instance, you wouldn't be able to assess the presence or absence of gene X if you were unaware that gene X even exists. As a consequence, this means at time step 0, the learner observes $d_0[\mathcal{X}^0]$—the projection of d_0 onto the variables in $\mathcal{X}^0$. Now suppose at time t, the learner discovers a new variable Z which they were previously completely unaware of, meaning $\mathcal{X}^t = \mathcal{X}^{t-1} \cup \{Z\}$. How should the learner adapt its current beliefs about the likely structure and parameters of the problem to accommodate this discovery?

The first problem is that the agent's current distributions over possible parent sets no longer cover all possible parent sets. For example, the current distribution over Pa_{X_1} does not include the possibility that Z is a parent of X_1. Worse, since the agent *cannot observe Z's past values* in $D_{0:t}$, it cannot observe $N^{t-k}_{Z=i|j}$, or $N^{t-k}_{X=i|j}$ when $Z \in Pa_X$ $(0 < k \leq t)$. The α-parameters involving Z are also undefined, and yet we need these values to calculate structure probabilities via (3.7) and parameters via (3.6) in the expanded hypothesis space.

Table 3.1 shows how introducing new variables makes the size of each (observed) state *dynamic*, in contrast to standard learning problems where they are static. We could try to phrase this as a missing data problem: Z was hidden in the past, but will be visible in future states, so why not estimate the likely value of the missing data using *structural expectation maximization* (Friedman, 1998)? However, such methods commit us to expensive passes over the full batch of data (thus losing *incrementality*) and also prevent us from decomposing the likelihood of a BN's structure as combination of local

parent-set probabilities.

Alternatively, we could just ignore states with missing information when counts involving the new variable are required. For example, we could use $P(Pa_{X_1}|D_{t:n})$ to score Pa_{X_1} when $Z \in Pa_{X_1}$ but use $P(Pa_{X_1}|D_{0:n})$ when $Z \notin Pa_{X_1}$. However, as Friedman and Goldszmidt (1997) point out, Bayesian scores like (3.1) assume we are evaluating models with respect to the *same data*. If two models are compared using different data sets (even if they come from the same underlying distribution), the learner tends to favour the model evaluated with the smaller amount of data (since we are typically unable to compute the normalising factor P(D)). As such, comparing two models with respect to different sets of data is an open problem.

Instead, our method discards the data $D_{0:t-1}$ gathered during the learner's previous deficient view of the hypothesis space, but *conserves* the relative *posterior* probabilities learned from the old, deficient view of the hypothesis space to construct a *prior* in the expanded hypothesis space.

3.2.1 Structure Priors

We start by defining the distribution over the expanded set of parent sets. Upon discovering the new variable Z at time t, the learner must update the distribution over the parent sets Pa_X for each $X \neq Z$ to include parent sets that include Z. Equation (3.8) constructs a new prior $P^t(Pa_X)$ using the old posterior (where C is a normalizing constant):

$$P^t(Pa_X) = \gamma P^t(Pa_X|\mathcal{P}_X) + (1-\gamma)P^0(Pa_X) \tag{3.8}$$

$$P^t(Pa_X|\mathcal{P}_X) = \begin{cases} 0 & \text{if } Pa_X \notin \mathcal{P}_X \\ \frac{(1-\rho)}{C}\mathrm{BD}_{t-1}(X, Pa_X) & \text{if } Z \notin Pa_X \\ \frac{\rho}{C}\mathrm{BD}_{t-1}(X, Pa'_X) & \text{if } Pa_X = Pa'_X \cup \{Z\} \end{cases} \tag{3.9}$$

This update preserves the relative likelihood among the parent sets that do not include Z. It is also maintains our bias towards simpler structures by re-assigning only a small proportion ρ of the probability mass of a parent set that does not include Z to the parent

set formed by adding Z to it. To make this concrete, imagine that before discovering Z, the agent believes $\{X_2\} \in \mathcal{P}_{X_1}$ and $BD_{t-1}(X_1, \{X_2\}) = 0.8$, with $\rho = 0.1$. Upon discovering Z, the agent will update its beliefs to $P^t(Pa_{X_1} = \{X_2\} | \mathcal{P}_{X_1}) = \frac{0.9*0.8}{C}$ and $P^t(Pa_{X_1} = \{X_2, Z\} | \mathcal{P}_{X_1}) = \frac{0.1*0.8}{C}$.

This still leaves us to define a distribution over Pa_Z—the possible parent sets of the newly discovered variable Z. The learner has no evidence on Z's parents at the moment of Z's discovery, so defaults to using the initial prior from (3.4).

3.2.2 Parameter Priors

To estimate the parameters θ^t, we return to the issue of the counts $N_{i|j}$ and the associated α-parameters. As before, we wish to avoid the complexity of estimating Z's past values. Instead, we *throw away* the past states $D_{0:t-1}$ and their counts $N^{t-1}_{i|j}$, but *retain* the relative likelihoods they gave rise to by packing these into updated values for the α-parameters, as shown in (3.10) (where K is a constant):

$$
\begin{aligned}
&N^t_{X=i|Pa_X=j} = 0 \text{ for all } i, j, X, Pa_X \\
&\alpha^t_{X=i|Pa_X=j} = \begin{cases} \frac{K}{|Z|} P(j|\theta^*_{t-1}) & \text{if } X = Z \\ \frac{K}{|Z|} P(i, j[Pa_X \setminus Z] | \theta^*_{t-1}) & \text{if } Z \in Pa_X \\ K * P(i, j | \theta^*_{t-1}) & \text{otherwise} \end{cases}
\end{aligned} \tag{3.10}
$$

Equation (3.10) summarizes the counts from states $D_{0:t}$ based on inferences from the previous best BN, then encodes these inferences in the α-parameters of the new parameter prior. Returning to the Asia example from figure 3.1, imagine we discover a new binary variable—*HIV*, which we think might affect a patient's chances of contracting tuberculosis, and have $K = 5$. We would then set $\alpha^t_{tub=1|asia=1,HIV=1}$ equal to $\frac{5}{2} * P(tub = 1, asia = 1) \mid \theta^*_{t-1})$. These revised α-parameters ensure that the likelihoods inferred from past states bias the estimated likelihoods of Pa and of θ when subsequent states arrive. Indeed, the larger K is, the more conservative the learner will be: i.e., the more the probability distribution it learned *before* discovering Z will influence its reasoning about dependencies and CPTs *after* discovering Z.

Algorithm 2 Learn BN in Expanding Hypothesis Space

function LEARN BN($\mathcal{X}^0$)
  Initialize $\mathcal{P}^0, Pa^0$ via Eq. (3.4)
  $N^0_{i|j} \leftarrow 0$ for all i, j, X, Π_X

  for $t = 0 \ldots maxTime$ **do**
    if new variable Z discovered **then**
      $\mathcal{X}^t \leftarrow \mathcal{X}^{t-1} \cup \{Z\}$
      $P^t(Pa_X) \leftarrow$ Update via (3.8) for all $Pa_X, X \in \mathcal{X}^{t-1}$
      $P^t(Pa_Z) \leftarrow$ Update via (3.4) for all Pa_Z
      $N^t_{i|j}, \alpha^t_{i|j} \leftarrow$ Eq. (3.10) for all i, j, X, Π_X
    else
      $\langle \mathcal{X}^t, \mathcal{P}^t, N^t_{i|j}, \alpha^t_{i|j} \rangle \leftarrow \langle \mathcal{X}^{t-1}, \mathcal{P}^{t-1}, N^{t-1}_{i|j}, \alpha^{t-1}_{i|j} \rangle$
    $d_t \leftarrow$ Generate new state
    $\langle N^t_{i,j}, \mathcal{P}^t, Pa^t, \theta^t \rangle \leftarrow$ UPDBN($\mathcal{P}^t, N^t_{i|j}, d_t$)
  return $\langle Pa^t, \theta^t \rangle$

3.2.3 Adapting the whole BN

Algorithm 2 outlines how the learner learns the structure and parameters of a BN incrementally, despite being initially unaware of the full hypothesis space. If the set of variables the learner is aware of remains stable, the learner updates its beliefs according to the latest trial as in Algorithm 1. However, if the learner discovers a new variable, then the learner uses equations (3.8) and (3.10) to conserve what it has learned so far while expanding its view of the hypothesis space.

We evaluate the accuracy of the learner's BN against the true BN by using a modified version of the *KL-Divergence* between the two distributions:

$$D^{\mathcal{X}}_{KL}(P \parallel Q) = \frac{1}{|v(\mathcal{X} \setminus \mathcal{X}^P)|} \sum_{s \in v(\mathcal{X})} P(s[\mathcal{X}^P]) \left(\log \frac{P(s[\mathcal{X}^P \cap \mathcal{X}])}{|v(\mathcal{X} \setminus \mathcal{X}^P)|} - \log \frac{Q(s[\mathcal{X}^Q \cap \mathcal{X}])}{|v(\mathcal{X} \setminus \mathcal{X}^Q)|} \right) \tag{3.11}$$

Equation (3.11) measures the similarity between distributions P and Q defined over the sets of variables $\mathcal{X}^P$ and $\mathcal{X}^Q$ respectively. The closer $D^{\mathcal{X}}_{KL}(P \parallel Q)$ is to zero, the

better the proposed distribution Q is at representing P. Unlike standard definitions of KL-divergence, (3.11) explicitly specifies $\mathcal{X}$—the set of variables defining the state-space one is measuring over. This is because, in our task, distributions P and Q may be defined over different sets of variables. Since the distributions do not distinguish between states where the values of a variable $X' \notin \mathcal{X}^P$ (or $X' \notin \mathcal{X}^Q$ respectively) differ, such differences must be marginalized out in the measurement.

If the interval between discovering new variables is large enough, the conservative priors created in Algorithm 2 should always provide a more accurate estimate of the true underlying distribution than a learner which abandons previously learned information and instead reverts to an uninformative prior upon discovering a new variable. Proposition 1 below formalizes this notion:

Let $\text{BN}^+ = P(\mathcal{X}^+|Pa^+,\theta^+)$ be the distribution over $\mathcal{X}^+$ given the true BN, and $\text{BN}^t = P(\mathcal{X}^t|Pa^t,\theta^t)$ be the distribution over $\mathcal{X}^t$ given the BN learned by an agent at time t following algorithm 2.

Further, let k be the interval between new variable discoveries. If the nth variable is discovered at time kn, let $\text{BN}^{kn}_{\neg con} = P(\mathcal{X}^{kn}|Pa^{kn}_{\neg con},\theta^{kn}_{\neg con})$ be the distribution over $\mathcal{X}^{kn}$ given the BN learned a *non-conservative* learner— one which, upon discovering a new variable, constructs new priors using (3.4), and by setting $N_{i|j} = 0$ and $\alpha_{i|j} = (|v(X \cup Pa_X)|)^{-1}$ for all X, Pa_X, i, j. The following proposition holds:

Proposition 1. *If* $D_{KL}^{\mathcal{X}^{kn}}(\text{BN}^+ \parallel \text{BN}^{kn+(k-1)}) \leq D_{KL}^{\mathcal{X}^{kn}}(\text{BN}^+ \parallel \text{BN}^{kn})$ *for* $n = 0,\ldots,N-1$, *then* $D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel \text{BN}^{kN}) \leq D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel \text{BN}^{kN}_{\neg con})$

Outline Proof. First note that for any $\mathcal{X}' \subseteq \mathcal{X}^+$, and any two distributions P_A and P_B defined over $\mathcal{X}'$, if $D_{KL}^{\mathcal{X}'}(\text{BN}^+ \parallel P_A) \leq D_{KL}^{\mathcal{X}'}(\text{BN}^+ \parallel P_B)$, then $D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel P_A) \leq D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel P_B)$.

Combined with our initial premise, this implies that the KL-divergence between BN^+ and BN^{kn} has been monotonically decreasing at each value of n, and therefore that $D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel \text{BN}^{kN-1}) \leq D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel \text{BN}^0)$.

Since $D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel \text{BN}^0) = D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel \text{BN}^{kn}_{\neg con})$ for any n, and since $D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel \text{BN}^{kn-1}) = D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel \text{BN}^{kn}_{con})$ via the definitions of (3.8) and (3.10), it immediately follows that $D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel \text{BN}^{kN}_{con}) \leq D_{KL}^{\mathcal{X}^+}(\text{BN}^+ \parallel \text{BN}^{kN}_{\neg con})$. □

The above proposition essentially states that (under certain conditions) conservativity

ensures that on discovering a new variable, you start to learn within the newly expanded hypothesis space with a bayesian network estimate that is closer to the true probabilistic belief model than if you don't adopt conservativity.

3.3 Experiments

Our experiments show that our method converges on an accurate model of the true BN's structure and parameters, even when the learner starts unaware of relevant variables. Further, by conserving probabilistic information as the hypothesis space expands, our learner maintains accurate models *throughout* learning, not just at the end of the learning process (as predicted by Proposition 1).

Our primary evaluation metric is the KL-Divergence as defined in (3.11) over $\mathcal{X}^+$ between the true distribution P^+ and the distribution P^t defined by the learner's current BN.

The other metric we use is to evaluate what the *BD-Score* (equation (3.1)) of the current learned BN *would* have been given the full batch of data $D_{0:T}$ and full awareness $\mathcal{X}^+$. This metric isolates how effective our algorithm is at discovering the inherent structure of the data in isolation from the learned parameters by assessing how a fully aware batch learner would rate the current BN.

We test our learner on four well-known BN learning problems: *Asia* (8 variables, 256 atomic states), *Sachs* (11 variables, 177147 atomic states), *Child* (20 variables, $\approx 10^9$ atomic states), and *Insurance* (27 variables, $\approx 10^{13}$ atomic states). [5] Each experiment runs over $10,000$ trials that are generated from the true BN. This is repeated 10 times; the results are averaged over these 10 simulations. For equations (3.4) and (3.8), we set $\rho = 0.1, \gamma = 0.99$, and for (3.10) we set $K = 5$. We set the maximum parent-set size to 4, and the time interval and tolerance threshold for the Buntine lattice update to $\tau = 100$ and $\kappa = 0.001$ respectively. When testing on a broad range of parameters, we found the above settings yielded good results (and that changing those parameters did not appear to alter the main findings reported below).

Our learner begins aware of just one variable (ie., $|\mathcal{X}^0| = 1$), and is then introduced to a new variable every 250 time steps (so the learner becomes fully aware of the entire

[5] Full specifications for each of the BNs above are available at `http://www.cs.huji.ac.il/~galel/Repository/`

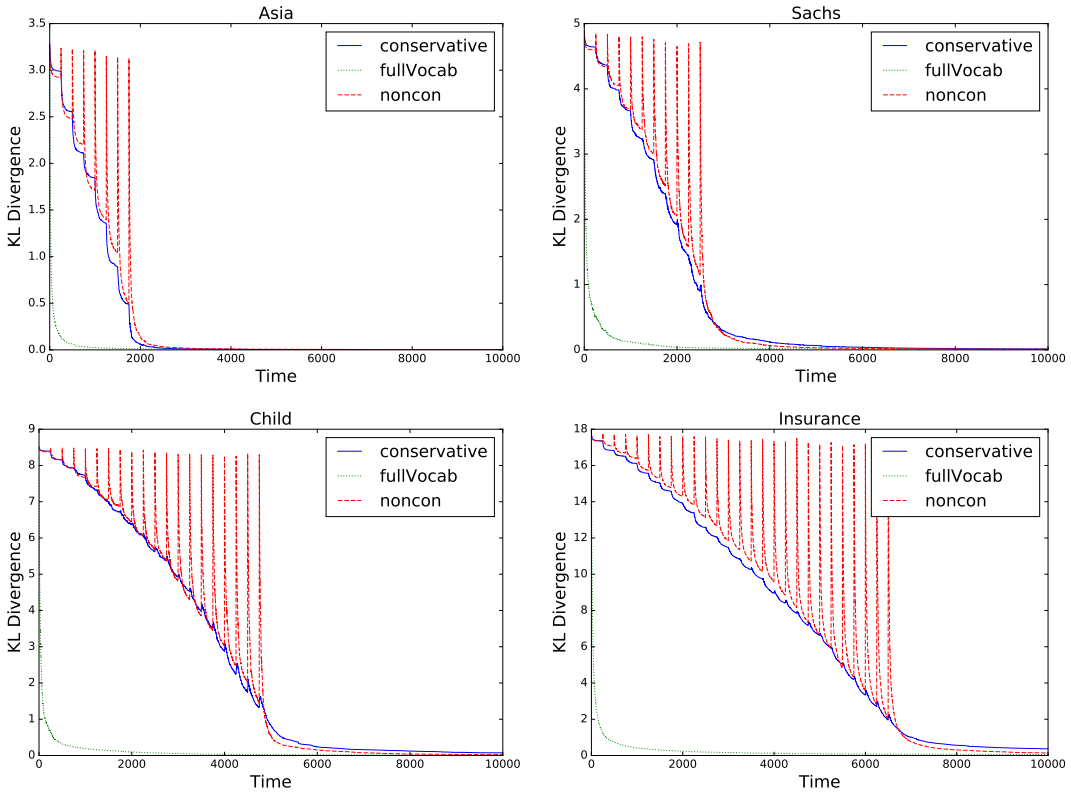


Figure 3.3: KL-Divergence of learned BNs from the true distributions

hypothesis space for *Insurance*, our largest BN, at state 6750).

We also compare our learner against two alternative models to test the effectiveness of our assumptions. The *non-conservative* learner does not use equations (3.8) or (3.10) to conserve information when a new variable is discovered, and instead discards previous states and the probability distribution over BNs learned from them. This alternative acts as a baseline model. Our upper-bound model, which we call the *full-vocab* learner, starts out fully aware of the hypothesis space (i.e., $\mathcal{X}^0 = \mathcal{X}^+$), but still has no knowledge of the true structure or parameters.

3.3.1 Results

Our learner was able to learn an accurate model of the underlying distribution of all 4 BNs, despite being initially unaware of all but one of the variables in the true hypothesis space. Figure 3.3 shows the KL-divergence between the true BN and the learner's BN at each time step. Notice that at any time step where a new variable is introduced, our learner's performance doesn't degrade while the non-conservative agent's performance

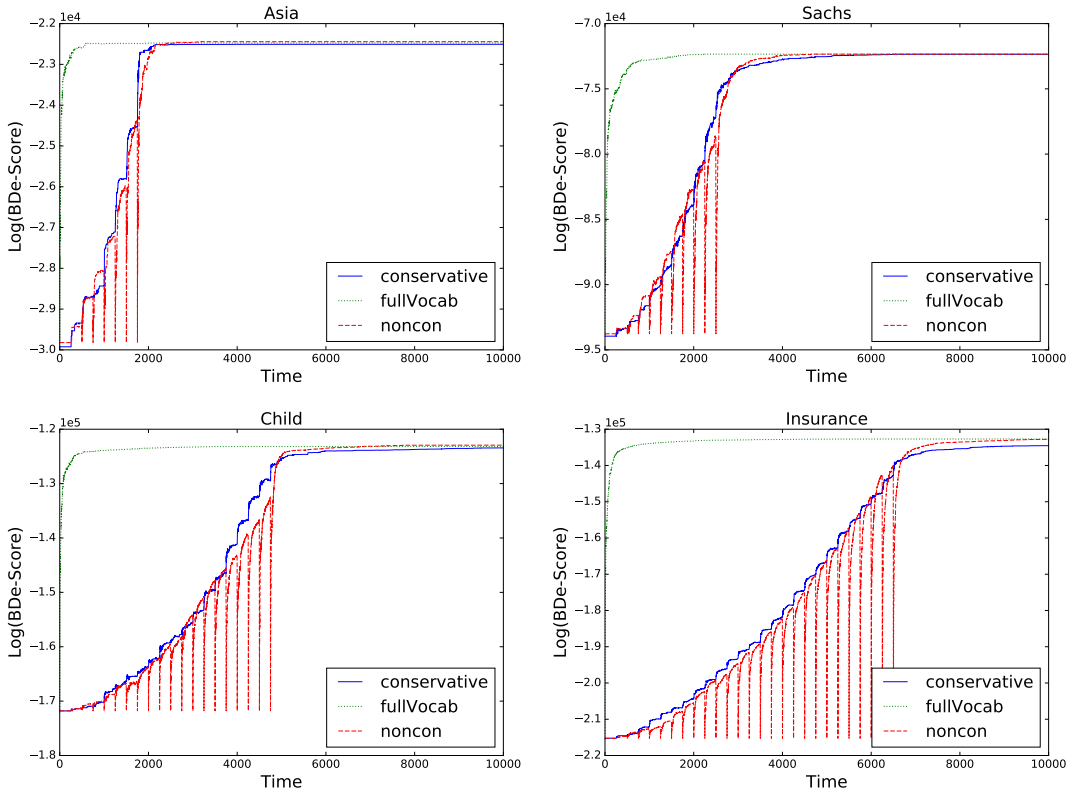
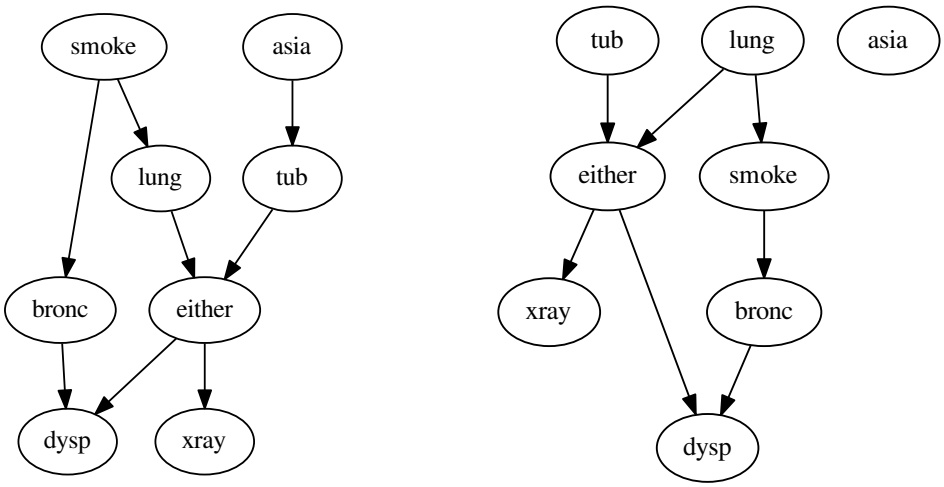


Figure 3.4: BD-Score of learned BNs

degrades considerably before improving again.

The same is also true of the BD-score, as shown in Figure 3.4. This indicates that the agent is able to learn a probabilistic structure which accurately reflects the underlying dependencies present in the data. To illustrate, figure 3.5 gives a typical example of the structure learned by the default agent on the *Asia* network. As can be seen, the learned structure is often almost identical to the true underlying BN, with a few minor exceptions. First, while the key probabilistic dependencies are captured, the learner is often unable to distinguish the causal directionality of dependencies from passive observation alone: for example, in figure 3.5b, there is a directed edge from lung cancer to smoking, rather than the other way around. Another difference is that the agent tends not to commit to dependencies for which there is too little data: as another example, in figure 3.5b, the agent does not commit to the dependency between visits to Asia and tuberculosis, as patients who have actually visited Asia are extremely rare ($P(Asia = true) = 0.01$).

One discrepancy to note is that, in the long period near the end of each experiment where the learner experiences a large number of states under the full hypothesis space, the non-conservative learner eventually achieves marginally better performance than the



(a) True BN

(b) Default agent learned BN

Figure 3.5: Typical structure learned by default agent on *Asia* task

conservative learner. For example, at $t = 10000$, the (log) BDe-Score of the conservative model is -1.345×10^5, while the non-conservative score is -1.328×10^5 (A difference which is statistically significant for $p = 1 \times 10^{-10}$). There are two likely reasons for this.

The first reason is that, given enough data, the data alone provides a fairly strong estimate of the structure and parameters of the true model. A learner with a highly concentrated prior which makes minor errors (i.e., the prior that the conservative learner constructs via (3.8)) will take longer to move towards a model favoured by the data than one with a prior which makes broad uniform predictions.

The second related issue is that while a prior with *structural modularity* like the BD-score (i.e., a structural prior which can be decomposed into the sum of priors of each variable's parent set) makes learning tractable, it potentially incorrectly penalizes structures which are actually quite close to the true BN. For example, consider an agent that is currently aware of two variables—X and Y—and thinks the mostly likely BN structure is $Y \to X$. Upon discovering a new variable Z, an agent using equation (3.8) to construct its new prior will initially rate the structure $Z \to Y \to X$ as higher than the structure $Y \to Z \to X$. In subsequent chapters, where the agent has direct control over action variables, or in sequential problems where the agent knows that one set of

variables always occurs before another set, such issues present less of a problem, as the agent has additional ordering information to guide its assessments.

3.4 Conclusion

In this chapter, we have provided a method to learn both the structure and parameters of a Bayesian Network, even when lifting the common assumption the learner is aware of all relevant belief variables in advance. Through experimentation on several well-known Bayesian Networks, we have demonstrated that by conserving previously learned information about structure and parameters, the learner is able to quickly converge on an accurate model of the problem upon discovery of new variables in the hypothesis space.

This chapter provides the groundwork for how to adapt a learner's existing beliefs once a new variable is discovered, but the explicit mechanisms of how such discoveries take place was left unspecified. In the next chapter we elaborate on how such discoveries can be made via *conversations with an expert* by specifying a dialogue protocol for two agents with differing levels of awareness about the underlying problem. Further, we expand our scope to problems where the agent must take actions within the environment, rather than just passively observing it.

Chapter 4

Structured Decision Problems with Unawareness

In this chapter, we move from models of belief to *decision tasks*, where the agent can take actions and receive rewards. Our agent learns how to behave optimally by learning a *Bayesian Decision Network* (DN): i.e., an extension of a Bayesian Networks with action and a utility (reward) nodes (see Section 4.1.1 for full definition). Once again, we start with an explanation of how the agent learns such models when it is fully aware of all relevant factors, then go on to describe how the agent can learn optimal behaviour when it starts *unaware* of the true set of belief and action variables, or the scope of the reward function.

The other main contribution of this chapter is to describe the mechanism by which the agent overcomes its own unawareness. In the previous chapter, the agent simply "became aware" of previously unforeseen variables at regular intervals during learning. In this chapter, we model a scenario closer to human teacher-apprentice learning, where an agent becomes aware of previously unforeseen actions and belief variables via discussion with an informed expert. This discussion uses an interaction policy with dialogue moves which would be quite natural in human teaching and learning environments.

A key point that is emphasised throughout this chapter is that we can now think of evidence from both domain trials and expert assistance as giving not only information about the relative counts of each state, but also *monotonic information* about the true DN in the form of a partial description. This means our agent must ensure that its current

model is valid in the sense that it is consistent with all such partial descriptions gathered so far. In the typical case where the agent is fully-aware of the hypothesis space, such validity is not much of a concern—the agent simply has to ensure that their current model is *well formed* (i.e., that their current model is a directed, acyclic graph). In the unaware case however, the agent must deal with validity in another sense— it must ensure that its current model is actually capable of producing the evidence it has seen so far. As we will see, if the current evidence is inconsistent with the agent's current estimate of the hypothesis space, the agent must detect this fact and fix it to overcome its own unawareness.

The focus of this chapter is on *single-step decision tasks*. That is, scenarios in which the agent takes an action based on some initial observations and immediately receives a final reward based on the outcome of this action. Any repetitions of the task are not dependent on one another. In the next chapter, we expand our model to handle *sequential decision tasks*, where the agent's actions in previous steps may affect the relative likelihood of outcomes of future actions, and hence also of potential future rewards.

4.1 Optimal Behaviour with Full Awareness

In a single stage decision problem, an agent observes some evidence e about its environment, takes some action a, then receives some immediate reward $\mathcal{R}(s)$ depending on the state s which resulted from their action.

Our agent's goal is to learn the policy π_+ which, given evidence e, always chooses the action a which maximizes its *expected utility* $EU(a|e)$:

$$EU(a|e) = \sum_{s'} P(s'|a,e)\mathcal{R}(s') \tag{4.1}$$

$$\pi_+(e) = \arg\max_{\pi} EU(\pi(e)|e) \tag{4.2}$$

If $P(s|a,e)$ or the reward function $\mathcal{R}$ are unknown, the agent must learn them via trial and error. Our agent can balance *exploiting* its current estimate of the best policy—π^*—with *exploring* the domain by using an ε-*greedy policy*. An ε-greedy policy is one

where, in all states, the agent acts randomly with probability $\varepsilon > 0$, and acts according to π^* with probability $1-\varepsilon$.

Further, for any policy π, we can measure the expected loss in reward against the true optimal policy π_+ using (4.3)—the *policy error*

$$Err(\pi) = \sum_e P(e)\left(EU(\pi_+(e)|e) - EU(\pi(e)|e)\right) \tag{4.3}$$

Unfortunately, if there are a large number of states, then evaluating the expected utility directly is often intractable. To make the problem tractable, we exploit the inherent structure of complex decision problems by learning a *bayesian decision network*.

4.1.1 Bayesian Decision Networks

A Bayesian Decision Network (DN) (Russell and Norvig, 2002) is a BN augmented with actions and rewards. It is a tuple:

$$\langle \mathcal{A}, \mathcal{X}, \mathcal{R}, Pa, \theta \rangle \tag{4.4}$$

Here, $\mathcal{A} = \{A_1, \ldots, A_m\}$ are the *action variables* the agent controls (where a full action a is a member of $v(\mathcal{A})$). As before, $\mathcal{X}$ are the belief variables, but they are now partitioned into $\mathcal{X} = \mathcal{B} \cup O$ ($\mathcal{B} \cap O = \emptyset$), where $\mathcal{B}$ is the set of variables the agent observes *before* taking action, and O are the variables which describe the *outcome* of that action. The reward function $\mathcal{R} : v(scope(\mathcal{R})) \rightarrow \mathbb{R}$ gives the reward received in each state, where $scope(\mathcal{R}) \subseteq \mathcal{X}$ are the variables which determine the reward.[1] The terms Pa and θ still describe the structure and effect of probabilistic dependencies (as in the definition of BNs), but now each Pa_X can also include both belief and *action variables* (i.e., $\forall X, Pa_X \subset \mathcal{X} \cup \mathcal{A}$).

As the agent takes actions within the domain, it gathers domain trials $D_{0:t} = \{d_0, \ldots, d_t\}$. A *domain trial* at time t is a tuple:

[1] To restrict the scope of this thesis, we assume that, given full knowledge of the true decision problem, the agent's preferences are *deterministic* on values of some subset of the outcome variables. We also assume the agent does not have a preference for particular actions, but only a preference for the *outcomes* that those actions bring about.

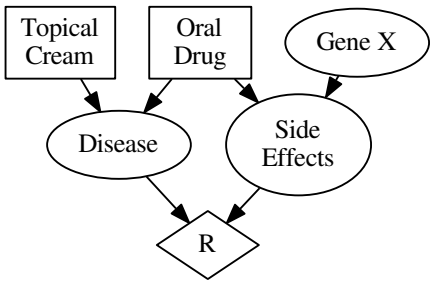


Figure 4.1: DN for medical example. Rectangles are action variables, ovals are belief variables, and diamonds are reward nodes

$$d_t = \langle s_t, a_t, r_t \rangle \tag{4.5}$$

Where $s_t = \langle b_t, o_t \rangle, b_t \in v(\mathcal{B}), o_t \in v(O), a_t \in v(\mathcal{A})$, and $r_t = \mathcal{R}(s_t)$.

Figure 4.1 shows an example DN. Here, the agent chooses assignments for action variables TOPICAL CREAM and ORAL DRUG based on observing GENE X, then receives a reward based on outcomes DISEASE, and SIDE EFFECTS. We note that DNs are essentially a subset of the notation used for *influence diagrams* (Howard and Matheson, 2005). Influence diagrams add *information arcs* from belief variables to action variables, which show the variables in $\mathcal{B}$ that the agent observes before choosing values for each $A \in \mathcal{A}$. For simplicity of explanation, we omit information arcs, and instead assume all action variables are set simultaneously, with access to observations of all variables in $\mathcal{B}$.

4.1.2 Learning the Probabilistic Dependencies

To learn the most likely probabilistic structure and parameters—Pa^* and θ^*, we can simply reuse the same technique of maximizing the ILP problem (3.5), and computing the expected parameters via (3.6) as in Chapter 3. The only differences are that we must add a few additional constraints to the types of structure that constitute a valid DN. For example, the action nodes are not permitted to have any parents, and the "before" nodes can only have other "before" nodes as parents (that is $\forall B \in \mathcal{B}, Pa_B \subset \mathcal{B}$).

4.1.3 Learning the Reward Function

Given full awareness of $\mathcal{X}$ and $scope(\mathcal{R})$, learning a valid reward function which conforms to the evidence seen in domain trials $D_{0:t}$ is relatively straightforward. For each state s_i that occurs in $D_{0:t}$, $\mathcal{R}_t(s_i) = r_i$. For unseen states, we default to indifference (e.g., by setting the reward of unseen states to 0). In the next section, we will see that when partial unawareness is introduced, this constraint solving problem becomes more complex.

Given a learned DN, the agent can then choose a policy π^* which maximizes the expected utility equation 4.1, where $a \in v(\mathcal{A})$ and $e \in v(\mathcal{B})$.

4.2 Overcoming Unawareness with Expert Guidance

So far, we've assumed our agent was aware of all relevant variables in $\mathcal{X} \cup \mathcal{A}$ and all members of $scope(\mathcal{R})$. We now drop this assumption. From here onward (as in the previous chapter), we denote the true set of belief variables, actions, and reward scope by $\mathcal{X}^+, \mathcal{A}^+$, and $scope_+(\mathcal{R})$; we denote the learner's awareness of them at time t by $\mathcal{X}^t$, $\mathcal{A}^t$, and $scope_t(\mathcal{R})$.

Suppose $\mathcal{X}^+ = \{X_1, X_2\}$, $X^0 = \{X_1\}$, $\mathcal{A}^+ = \{A_1, A_2\}$, $\mathcal{A}^t = \{A_1\}$. We assume the agent can't observe the value of variables it is unaware of. Additionally, we assume that an agent cannot observe the value of a variable at a past time at which it was unaware of it, even after becoming aware of it. In the medical example from the introduction, if X_2 is particular gene, we assume the agent cannot detect the presence of that gene if it is unaware it exists. The agent also cannot have preferences that are dependent on the gene: there is no *de-re* preference without *de-re* belief (Cadilhac *et al.*, 2015). We also assume the agent cannot perform actions it is unaware of (i.e., if the agent is unaware of variable A_2 then it cannot set it to a value other than its default value $A_2 = 0$).[2] This means at time $t = 0$, the agent does not observe the true trial d_0, but rather, its projection onto $\mathcal{X}^0$ and $\mathcal{A}^0$:

$$\langle s_0[\mathcal{X}^0], a_0[\mathcal{A}^0], r_0 \rangle \tag{4.6}$$

[2]This assumption, while reasonable, may not always hold (E.g., an agent may accidentally lean on a button while unaware that the button is part of the task).

But awareness of those missing factors may be crucial to learning π_+. For example, the best action may depend upon whether X_2 is true, or the optimal policy may sometimes involve setting A_2 to a value other than its default.

The following sections thus answer two main questions. First, how can an agent discover and overcome its own unawareness? Second, when an agent discovers a new variable, how can they integrate it into their current model while conserving what they have learned from past experience?

Overcoming unawareness will be down to a combination of two factors: certain inferences on the part of the learner, which can be summarised roughly as "I'm stuck", and which prompt a question whose possible answers will help the learner overcome their current dilemma, and unsolicited advice from the expert, which in turn occurs when the learner has been performing the task sufficiently poorly. These are quite natural kinds of moves one would expect in a scenario where a teacher is interacting with an apprentice, as the apprentice attempts domain-level actions to solve their task.

4.2.1 Expert Guidance

Our agent can expand its awareness via advice from an expert. Teacher-apprentice learning is common in the real world, as it allows learners to receive contextually relevant advice which may inform them of unforeseen concepts.

This thesis assumes the expert is cooperative, sincere and infallible. Further, we abstract away from the complexity of grounding natural language statements in a formal semantics and instead assume that the agent and expert communicate via a pre-specified formal language (though see e.g., Zettlemoyer and Collins (2007) for work on this problem). We do not, however, assume the expert knows the *agent's* beliefs about the DN.

The formal language used for communication is effectively one in which each well-formed formula δ acts as a partial description of the true DN, dn_+ (formally, $dn_+ \models \delta$). In other words, when the agent asks a question, the answer will be a partial description of the true DN, and when the expert gives unsolicited advice, this will also be a partial description of the true DN. The model theory of this language is given in the Appendix: roughly speaking, each model corresponds to a unique DN, and a model satisfies a formula of this language if and only if that formula is a partial description of the DN.

We use the variables $\mathcal{X}^e$ and $\mathcal{A}^e$ to express the expert's current knowledge of our agent's awareness (where by "knowledge", we mean that the expert has concrete and indefeasible evidence that the agent is aware of the variable; the expert does not build a belief model of the agent's awareness, probabilistic or otherwise). An action A only becomes a member of $\mathcal{A}^e$ if the expert has observed the agent setting A to a non-default value in a past trial, or if the expert has explicitly mentioned A to the agent in a previous piece of advice. Likewise, a variable X only becomes a member of $\mathcal{X}^e$ when either the expert or agent has explicitly mentioned it in a previous piece of dialogue.

As argued in Chapter 1, the goal is to provide a minimal set of communicative acts so that interaction between the agent and expert resembles dialogue moves that would be quite natural in a human teacher-apprentice interaction.

Concretely, this means we want our system to have two properties. First, the expert should mostly allow the agent to learn by themselves, interjecting only when the agent performs sufficiently poorly, or when they explicitly ask a question. Second, our expert should be able to give *non-exhaustive* answers to queries. This is because, in practice, it is unlikely a human expert will be able to give exhaustive answers to all questions due to either cognitive bounds or a lack of information. Following the maxims of Grice (1975), a cooperative speaker should give answers which provide just enough information to resolve the agent's current dilemma.

We identify four types of advice, whose combination guarantee the agent eventually behaves optimally, regardless of initial awareness.

4.2.1.1 Better Action Advice

If the expert sees the agent perform a sub-optimal action, it can tell the agent a better action to take instead. For example: "When it is raining, take your umbrella instead of your sun hat". Our goal is to avoid interrupting the agent every time it makes a mistake, so we specify the following conditions for when the agent performs poorly enough to warrant correction: Let t be the current time step, and t' be the time the expert last uttered advice. When (4.7-4.11) hold:

$$t - t' > \mu \tag{4.7}$$

$$\sum_{t' \leq i \leq t} \frac{EU(\pi_+(b_i)|b_i) - EU(a_i|b_i)}{t - t'} > \beta \tag{4.8}$$

$$EU(a_t|b_t) \geq r_t \tag{4.9}$$

$$\exists \mathcal{A}', a' \in v(\mathcal{A}^e \cup \mathcal{A}'), EU(a'|b_t) > EU(a_t|b_t) \tag{4.10}$$

$$\begin{aligned} \nexists \mathcal{A}'', (a'' \in v(\mathcal{A}^e \cup \mathcal{A}''), EU(a''|b_t) > E(a_t|b_t) \\ \wedge |\mathcal{A}''| < |\mathcal{A}'|) \end{aligned} \tag{4.11}$$

Then the expert will utter advice of the form (4.12):

$$EU(a'|w_t^b) > EU(a_t[\mathcal{A}^t \cup \mathcal{A}']|w_t^b) \tag{4.12}$$

Equation (4.7) ensures some minimum time μ has passed since the expert last gave advice, while (4.8) ensures the expert won't interrupt unless its estimate of the agent's policy error is above some threshold β. Together, μ and β define how *tolerant* the expert will be to the agent's poor performance before deciding to give advice. Equation (4.9) ensures that the expert's suggested action has an expected reward higher than what the agent received at time t. Finally, (4.10-4.11) ensure not only that a better action a' exists, but also that a' introduces the minimum amount of potentially unforeseen action variables $\mathcal{A}'$ needed to improve the agent's behaviour. This means there isn't another action a'' which could be advised while revealing fewer unaware variables to the agent. This follows from our desire to give non-exhaustive advice, by first suggesting improvements which use concepts the agent is already aware of, rather than introducing many new action variables all at once.

Equation (4.12)—the expert's utterance—requires explanation. On first thought, the expert should utter:

$$EU(a'|b_t) > EU(a_t|b_t) \tag{4.13}$$

which fully describes b_t. But remember that the agent's awareness $\mathcal{B}^t$ may be a tiny subset of $\mathcal{B}^+$. Uttering such advice may involve revealing many variables the agent is currently unaware of. This is exactly the type of cognitively-taxing exhaustive explanation we wish to avoid.

Alternatively, the expert could restrict the description to $\mathcal{X}^e$, but this could make the advice false: there may exist a $b' \neq b_t$ where $b'[\mathcal{X}^e] = b_t[\mathcal{X}^e]$, but $EU(a_t|b') > EU(a'|b')$.

To address this issue, the expert uses a *sense-ambiguous term* w^b, whose *intended* meaning is the true state b (i.e $[\![w^b]\!] \in v(\mathcal{B}^+)$, where $[\![y]\!]$ is the *denotation* of y), but whose *default* interpretation by the agent is $b[\mathcal{B}^t]$. In words, the expert says *"In that last situation, a' would have been a better action than a_t"*, which implicitly makes reference to the state of the world in the previous time step.

By introducing ambiguity, the agent can now interpret (4.12) in two ways. The first is as a *partial description* of the true DN (dn_+), which is true *regardless* of what it learns in future. More formally, if δ is the partial description given by the current piece of dialogue (or domain) evidence, then $dn_+ \models \delta$. [3]

On hearing (4.12), the agent adds (4.14-4.15) to its list of partial descriptions of the true DN:

$$\forall A \in \mathcal{A}', A \in \mathcal{A}^+ \wedge \exists X \in scope_+(\mathcal{R}), anc(A,X) \tag{4.14}$$

$$\exists s, s[\mathcal{B}^t] = s_t[\mathcal{B}^t] \wedge \mathcal{R}_+(s) > r_t \tag{4.15}$$

Where $anc(X,Y)$ means X is the ancestor of Y (i.e., there is a directed path from X to Y in the true DN):

$$anc(X,Y) = X \in Pa_Y \vee (\exists Z, X \in Pa_Z \wedge anc(Z,Y)) \tag{4.16}$$

Equations (4.14-4.15) imply that any variable the expert utters must be *relevant* to the problem. In other words, it must exert influence on at least one variable in $scope_+(\mathcal{R})$, and that there exists some state the agent could have reached which would yield a better reward than the one it got. In fact, we can enforce all equations of the form (4.14) when learning Pa^* by adding the constraint (4.17) to our linear program:

$$\forall Y \notin scope(\mathcal{R}) : \sum_{\substack{X \in \mathcal{X} \\ Pa_X : Y \in Pa_X}} I(Pa_X \rightarrow X) \geq 1 \tag{4.17}$$

[3] A full specification of syntax and semantics used to partially describe DNs is given in the appendix

The second way the agent could use (4.12) is by adding its *default interpretation* of the advice to its current knowledge:

$$\forall b \in \mathcal{B}^+ : b[\mathcal{B}^t] = b_t[\mathcal{B}^t] \Rightarrow EU(a'|b) > EU(a|b) \tag{4.18}$$

The agent can then enforce (4.18) by choosing a' whenever $b[\mathcal{B}^t] = b_t[\mathcal{B}^t]$, regardless of what seems likely from $D_{0:t}$. We should now see that even with a cooperative, infallible expert, even abstracting away issues of grounding natural language, misunderstandings can still happen due to differences in agent and expert awareness. As the next section shows, such misunderstandings can reveal gaps in the agent's awareness and help to articulate queries whose answers guarantee the agent expands its awareness.

Lemma 1 guarantees the expert's advice strategy continues to reveal unforeseen actions to the agent so long as its performance in trials exceeds the expert's tolerance.

Lemma 1. *Consider an agent with awareness $A^t \subset A^+$, and expert following (4.7-4.12). As $k \to \infty$, either $Err(\pi_{t+k}) \to c \leq \beta$ or the expert utters (4.12) where $\mathcal{A}' \neq \emptyset$.*

Proof. For finite μ, $(t+k) - t' > \mu$ as $k \to \infty$, satisfying (4.7).

Given $\mathcal{A}^t$ remains fixed, the agent's policy will eventually converge to some policy π. If $Err(\pi) < \beta$, we are done. If $Err(\pi) \geq \beta$, then (4.8) will be satisfied.

Since the agent is ε-greedy, at every time step it has a non-zero probability of performing all actions $a \in v(\mathcal{A}_t)$, meaning that eventually $\mathcal{A}^e = \mathcal{A}^t$. Further, since $\pi \neq \pi_+$, there must exist some $b \in v(\mathcal{B}^+)$ and $a' \in v(\mathcal{A}^+)$ such that $\forall a \in \mathcal{A}^t, EU(a'|b) > EU(a|b)$, thereby satisfying (4.10).

Since the agent is ε-greedy, it is guaranteed to eventually perform $a^* = \arg\max_{a \in v(\mathcal{A}^t)} EU(a|b)$ in in state b, thus making (4.11) true for a non-empty value of $\mathcal{A}'$.

(4.7-4.11) are not mutually exclusive, so all four will eventually be true at once, causing the expert to utter (4.12) for non-empty $\mathcal{A}'$. □

4.2.1.2 Resolving a Misunderstanding

We noted before that the agent's default interpretation of (4.12) could lead it to misunderstand the expert's intended meaning. To illustrate, suppose the agent receives advice (4.19) and (4.20) at times $t-k$ and t:

$$EU(a|w^b_{t-k}) > EU(a'|w^b_{t-k}) \tag{4.19}$$

$$EU(a|w^b_t) < EU(a'|w^b_t) \tag{4.20}$$

While the intended meaning of each statement is true, the agent's default interpretations of w^s_{t-k} and w^s_t may be identical. That is, $b_{t-k}[\mathcal{B}^t] = b_t[\mathcal{B}^t]$. From the agent's perspective, (4.19) and (4.20) conflict, and thus give the agent a clue that its current awareness of $\mathcal{X}^+$ is deficient. To resolve this conflict, the agent asks (4.21) (in words, *"which B has distinct values in b_{t-k} and b_t?"*) and receives an answer which provides monotonic information of the form (4.22-4.23):

$$?\lambda B(B \in \mathcal{B}^+ \wedge s_{t-k}[B] \neq s_t[B]) \tag{4.21}$$

$$B' \in \mathcal{B}^+ \tag{4.22}$$

$$B' \notin scope_+(\mathcal{R}) \implies \exists X \in scope_+(\mathcal{R}), anc(B', X) \tag{4.23}$$

Notice there may be many variables in $\mathcal{B}^+ \setminus \mathcal{B}^t$ whose assignments differ in b_{t-k} and b_t. Thus, the expert's answer can be *non-exhaustive*. This means the agent must abandon its previous defeasible interpretations of the form (4.18), but can keep (4.14-4.15), as these are true regardless of what the true DN is. Lemma 2 guarantees the expert will eventually reveal new belief variables as long as the agent's current awareness is such that misunderstandings are still possible.

Lemma 2. *Consider an agent with awareness $\mathcal{X}^t \subset \mathcal{X}^+$, $\mathcal{A}^t = \mathcal{A}^+$. If $\exists b', \exists b \neq b', b[\mathcal{B}^t] = b'[\mathcal{B}^t]$ and $\pi_+(b) \neq \pi_+(b')$, then as $k \to \infty$, either $Err(\pi_{t+k}) \to c \leq \beta$, or the expert utters (4.22) such that $B' \notin \mathcal{B}^t$*

Proof. If the agent converges to some policy π such that $Err(\pi) \leq \beta$, we are done.

Assume $a = \pi_+(b)$ and $a' = \pi_+(b')$. Consider that for all k, $P(b_{t+k} = b) > 0$, and (since the agent is ε-greedy) $P(\pi_{t+k}(b) = a') > 0$, where $a' = \pi_+(b')$.

If $Err(\pi) \geq$, then at some time $t + k_1$ where $(t + k_1) - t' > \mu$ and $b_{t+k_1} = b$ the agent will perform a', thus satisfying (4.7-4.11) and causing the expert to utter $EU(a|w^b_{t+k_1}) > EU(a'|w^b_{t+k_1})$.

By similar reasoning, at some time $b_{t+k_2} = b'$ the expert will utter $EU(a', w^b_{t+k_2}) > EU(a, w^b_{t+k_2})$.

Under $\mathcal{B}^t$, $[\![w^s_{t+k_1}]\!] = [\![w^s_{t+k_2}]\!]$, so the agent will ask (4.21) with answer $B \notin \mathcal{B}^t$. □

4.2.1.3 Unforeseen Rewards

In typical DNs (where we assume the agent starts fully aware of $\mathcal{X}^+, A^+$, and $scope_+(\mathcal{R})$), we tend only to think of the trials as providing *counts*. For an unaware agent, a trial $d_t = \langle s_t, a_t, r_t \rangle$ also encodes *monotonic information*:

$$\exists s, s[\mathcal{X}^t] = s_t[\mathcal{X}^t] \wedge \mathcal{R}_+(s) = r_t \tag{4.24}$$

This, along with (4.15), constrain the form of $\mathcal{R}$ the agent can learn. Recall that $scope_t(\mathcal{R})$ may be only a subset of $scope_+(\mathcal{R})$, so it might be impossible to construct an $\mathcal{R} : v(scope_t(\mathcal{R})) \to \mathbb{R}$ satisfying all descriptions of the form (4.24) and (4.15) gathered so far. Further, those extra variables in $scope_+(\mathcal{R}) \setminus scope_t(\mathcal{R})$ may not be in $\mathcal{X}^t$. To resolve this, if the agent fails to construct a valid $\mathcal{R}$, it asks (4.25) (in words, *"which variable X (that I don't already know) is in scope*$(\mathcal{R})$*?"*), receiving an answer of the form (4.26):

$$?\lambda X(X \in scope_+(\mathcal{R}) \bigwedge_{X' \in scope_t(\mathcal{R})} X \neq X') \tag{4.25}$$

$$X'' \in scope_+(\mathcal{R}) \wedge X'' \in \mathcal{X}^+ \tag{4.26}$$

Again (4.26) may be *non-exhaustive*. Even so, Lemma 3 guarantees that the agent's reward function eventually equals $\mathcal{R}_+$.

Lemma 3. *Consider an* ε*-greedy agent with awareness* $\mathcal{A}^t = \mathcal{A}^+$, $scope_t(\mathcal{R}) \subseteq scope_+(\mathcal{R})$. *As* $k \to \infty$, *there exists a K such that* $\forall s, \forall k \geq K$, $\mathcal{R}_{t+k}(s) = \mathcal{R}_+(s)$.

Proof. Since $\mathcal{A}^t = \mathcal{A}^+$, and the agent is ε-greedy, then over infinite time the agent will eventually enter s at some time i, receiving reward $\mathcal{R}_+(s)$, and update its current reward function so that $\mathcal{R}_i(s) = \mathcal{R}_+(s)$. If the agent has previously encountered another s' such that $s[scope_t(\mathcal{R})] = s'[scope_t(\mathcal{R})]$ and $\mathcal{R}_+(s) \neq \mathcal{R}_+(s')$, the partial descriptions (4.24) for s and s' will conflict. The agent resolves this by asking (4.25), receiving an answer differentiating s from s' in $\mathcal{R}_+$. □

4.2.1.4 Unknown effect

Recall that, to keep the problem tractable, our agent searches for DNs in the space of "reasonable" parent sets $\mathcal{P}$. Unfortunately, there might be no valid DN within $\mathcal{P}$ which also satisfies the constraints of the form (4.17). The most obvious case of this is when a variable $X \notin scope_t(\mathcal{R})$ has no children in *any* reasonable DAG (i.e., $\forall Y, \forall Pa_Y \in \mathcal{P}_Y, X \notin Pa_Y$).[4] Here, the agent can ask *why X* is relevant by asking (4.27) (*"what V does X affect directly?"*):

$$?\lambda V(X \in Pa_V) \tag{4.27}$$

In response, the expert answers with the name of a variable V' which X directly affects:

$$X \in Pa_{V'} \tag{4.28}$$

Which imparts the monotonic partial description (4.29-4.31) to the agent:

$$V' \in \mathcal{X}^+ \tag{4.29}$$

$$X \in Pa_{V'} \tag{4.30}$$

$$V' \notin scope_+(\mathcal{R}) \implies \exists Y \in scope_+(\mathcal{R}), anc(V', Y) \tag{4.31}$$

This advice guarantees that the agent which prunes the set of reasonable parents for each variable will always be able to construct a valid DN, provided its list of reasonable parents obey the partial descriptions given by (4.30).

4.3 Adapting to Advice with Unforeseen Factors

Section 4.2.1 showed four ways to expand its awareness of $\mathcal{X}$, $\mathcal{A}$, and $scope(\mathcal{R})$. To improve on simply restarting learning, we must now say how the agent *adapts* its beliefs about Pa and θ when its awareness expands.

[4] More generally, this can occur whenever there is no directed path from X to $scope(\mathcal{R})$ via the union of all members of $\mathcal{P}$

4.3.1 Adding a new belief variable

As covered in Chapter 3, when the agent discovers a new belief variable Z at time t, its old distributions over parent sets no longer cover all possible parents. Further, the agent cannot go back and observe Z's past values in $D_{0:t-1}$. Our approach to handling this issue remains unchanged from Chapter 3—the agent creates new priors for Pa and θ in the expanded space using equations (3.8) and (3.10) respectively.

4.3.2 Adding a new action variable

When the agent discovers a new action variable A, the same issue arises—the agent did not consider A as a possible parent to any $X \in \mathcal{X}^{t-1}$, so must revise $\mathcal{P}$ with A as a possibility. Unlike for belief variables, however, we don't need to discard $D_{0:t-1}$. Since we assume that the agent cannot influence action variables it is unaware of, we can simply fill in the default value of A in all past trials $D_{0:t-1}$.

4.3.3 Expanding the reward function scope

Learning that a variable X is in the scope of $\mathcal{R}$ may cause us to revise Pa^*, even if $X \in \mathcal{X}^{t-1}$. This is because expanding $scope_t(\mathcal{R})$ loosens the constraints of the form (4.17) which may allow us to construct a higher scoring DN structure that was previously invalid.

4.3.4 The Overall Learning Process

Algorithm 3 outlines the entire learning process described throughout the previous sections. In most steps, the agent incrementally revises its model based on the latest trial and current reasonable parents. If enough time τ passes, or the agent's awareness expands, the agent makes larger changes to its model, including revising $\mathcal{P}$.

Given algorithm 3, Theorem 1 guarantees our agent is eventually able to learn all the action and state variables which are relevant to expressing an optimal policy, regardless of initial awareness. Further, if the structure learning stage does not prune true dependencies from $\mathcal{P}$ (which we can ensure by setting $\kappa = 0$), then the agent is guaranteed to

Algorithm 3 Learning DNs with Unawareness

Input: $\mathcal{A}^0$, $\mathcal{X}^0$, Pa^0, θ^0, $\mathcal{P}^0$
for $t = 1 \ldots maxTrials$ **do**
  $b_t \leftarrow$ Generate new state
  $\langle o_t, a_t, r_t \rangle \leftarrow \varepsilon$-GREEDY$(b_t[\mathcal{X}^t])$
  $N^t_{i|j} \leftarrow$ Update with $b_t[\mathcal{X}^t], o_t[\mathcal{X}^t]$ for all i, j
  $BD_t(X, Pa_X) \leftarrow$ Update via (3.7) for $Pa_X \in \mathcal{P}_X$
  if $t \equiv 0 \pmod{\tau}$ **then**
    Revise $\mathcal{P}^t$ via (Buntine, 1991) lattice update
  if Exists X with no path to reward via $\mathcal{P}^t$ **then**
    Ask (4.27) and update $\mathcal{X}^t$, $\mathcal{P}^t$
  $\mathcal{R}^t \leftarrow$ Update with constraints (4.24, 4.15)
  if Update to $\mathcal{R}_t$ fails **then**
    Ask (4.21) and update $scope_t(\mathcal{R}), \mathcal{X}^t, \mathcal{P}^t$
  if (4.7-4.11) are true **then**
    actAdvice $\leftarrow$ Expert advises (4.12)
    Update $\mathcal{A}^t$, $\mathcal{P}^t$ according to *actAdvice*
    if *actAdvice* conflicts with past utterances **then**
      Ask (4.21) and update $\mathcal{X}^t, \mathcal{P}^t$
  if $\mathcal{X}^{t-1} \neq \mathcal{X}^t$ **then**
    Update $N^t_{i|j}, \alpha_{i|j}, \mathcal{P}, P(Pa)$ via (3.4, 3.8, 3.10)
  $\langle Pa^t, \theta^t \rangle \leftarrow$ Solve (3.5) (with (added constraints 4.17)), and (3.6)

converge on a policy which is within some bound of the true optimal policy for the task. This bound is dictated by β - the expert's tolerance for the agent's mistakes.

Theorem 1. *Consider an agent with initial awareness* $\mathcal{X}^0 \subseteq \mathcal{X}^+, A^0 \subseteq A^+, scope_0(\mathcal{R}) \subseteq scope_+(\mathcal{R})$ *following algorithm 3 (with* $\kappa = 0$*). As* $t \to \infty$*,* $Err(\pi_t) \to c \leq \beta$.

Proof. By repeatedly applying Theorems 1-3, either $Err(\pi_t) \to c \leq \beta$ (in which case we are done), or there exists a K where $\mathcal{A}^K = \mathcal{A}^+$, $\mathcal{R}_K = \mathcal{R}_+$, and $\mathcal{X}^K = \mathcal{B}^k \cup O^K$. Here, $\mathcal{B}^K$ contains all variables $\mathcal{B} \in \mathcal{B}^+$ such that there exist b, and b' where $b[\mathcal{B}^+ \setminus B] = b'[\mathcal{B}^+ \setminus B]$ but $b[B] \neq b'[B]$ and $\pi_+(b) \neq \pi_+(b')$. In other words, $\langle \mathcal{X}^K, \mathcal{A}^K, \mathcal{R}^K \rangle$ define a related decision problem with an identical optimal policy and marginal probability distribution to the original problem, but for which the agent is fully aware.

The BD-score is a *consistent score*, which means that as $|D| \to \infty$, Pa^* contains all dependencies present in the true distribution, regardless of initial prior (assuming we have not over-pruned the search space, which is true in this case because $\kappa = 0$). As a result, as $t \to \infty$, then $\forall a \in v(\mathcal{A}^+)$, $\forall b \in \mathcal{B}^K$, $\forall y \in v(scope_+(\mathcal{R}))$, we have that $P(y|a,b,Pa_t^*,\theta_t^*) \to P(y|a,b,Pa^+,\theta^+)$.

Since our agent's probabilistic model converges to the true probabilistic distribution, and because $\mathcal{R}^K = \mathcal{R}^+$ and $\mathcal{A}^K = \mathcal{A}^+$, we have that $\pi_t^* \to \pi_+$ and therefore that $Err(\pi_t^*) \to 0$ □

We can also guarantee that an agent following algorithm 3 produces a DN at each time step which is consistent with all partial descriptions gathered from evidence so far:

Theorem 2. *Let $\delta_{0:t}$ be the conjunction of all partial descriptions gathered up to time step t. An agent following algorithm 3 is guaranteed to produce a* DN *at time step t such that $dn_t \vDash \delta_{0:t}$.*

Outline proof. $\delta_{0:t}$ contains four kinds of conjuncts:

(i) $X \in \mathcal{X}^+, X \in \mathcal{A}^+, X \in \mathcal{B}^+, X \in O^+, X \in scope_+(\mathcal{R})$

(ii) $X \in Pa_Y$

(iii) $\exists s, s[Y] = s_i[Y]$ such that either $\mathcal{R}_+(s) > r_i$ (as in equation (4.15)) or $\mathcal{R}_+(s) = r_i$ (as in equation (4.24))

(iv) $\exists Y \in scope_+(\mathcal{R}), anc(X,Y)$

Lines 10, 13, 16, and 18 ensure that conjuncts of type (i) are satisfied, as the agent immediately updates the sets $\mathcal{X}^t, \mathcal{A}^t, \mathcal{B}^t, O^{\sqcup}$, and $scope_t(\mathcal{R})$ on receipt of such evidence.

Each conjunct of type (ii) is satisfied by directly including a constraint of the form $\sum_{Pa_Y : X \in Pa_Y} I(X \to Pa_Y) = 1$ in the ILP-maximization at step 21. Alternatively, we can satisfy this constraint earlier by setting the probability of any Pa_Y for which $X \notin Pa_Y$ to 0 in steps 6 and 8.

Conjuncts of type (iii) are explicitly used as constraints in computing a valid reward function in step 11. If it is not possible for the agent to generate a valid reward function satisfying all type (iii) constraints given its current view of the hypothesis space, then the evidence received in step 13 will ensure the hypothesis space expands in such a way as to guarantee a valid reward function (see Lemma 3).

Conjuncts of type (iv) are satisfied in the ILP-step. The combination of the three constraints listed in equation (3.5) guarantee the agent will output a parent structure which is a directed-acyclic graph (see Bartlett and Cussens (2017) for details). The additional constraint (4.17) ensures that every variable which is not in the scope of the reward function must have at least one child. By definition, if a directed graph is acyclic and every non-reward node has at least one child node, then there must exist a path from every non-reward node to at least one reward node, thus satisfying all conjuncts of type (iv). Further, if it is not possible to construct such a graph given the current set of reasonable parents $\mathcal{P}$, then the agent will ask 4.27, whose answer will provide the name of a variable which is the child of a (currently childless) variable. □

4.4 Experiments

Our experiments show that agents following algorithm 3 converge to true optimal policy in practice, even when using a heuristic which attempts to minimize complexity. We also show that conserving information improves results. We do not investigate assigning explicit costs to agent-expert communication, but do show how varying the expert's tolerance affects the agent's performance.

We tested agents on two different types of task. The first was a manually constructed decision network called the *barley* problem (inspired by Kristensen and Rasmussen (2002)). This is a decision problem modelled on crop-farming, in which the agent must decide on factors such as which grain to plant and how much fertiliser to use, with the goal of producing a harvest with high yields of healthy (high protein) crops.[5] Figure 4.2 shows an illustration of the barley DN. In each experiment, our agent begins with a simplified awareness of the problem: $\mathcal{X}^0 = \{Yield, Protein, Gross\ Crops, Nitrogen, Precipitation, Soil\ Type\ \}$, $\mathcal{A}^0 = \{Grain, Fertiliser\}$, and $scope_0(\mathcal{R}) = \{Yield, Protein\}$. In other words, the agent starts out unaware of 56% of the factors that define the optimal policy in this domain. This task was explicitly designed such that the agent cannot achieve the optimal policy without considering factors which it is initially unaware of. For instance, in hot weather, the agent should apply fungicide to crops to prevent fungus outbreaks, but the agent is initially unaware that the concept of fungus is even an issue.

[5] Full specifications for all DNs are included in the appendix

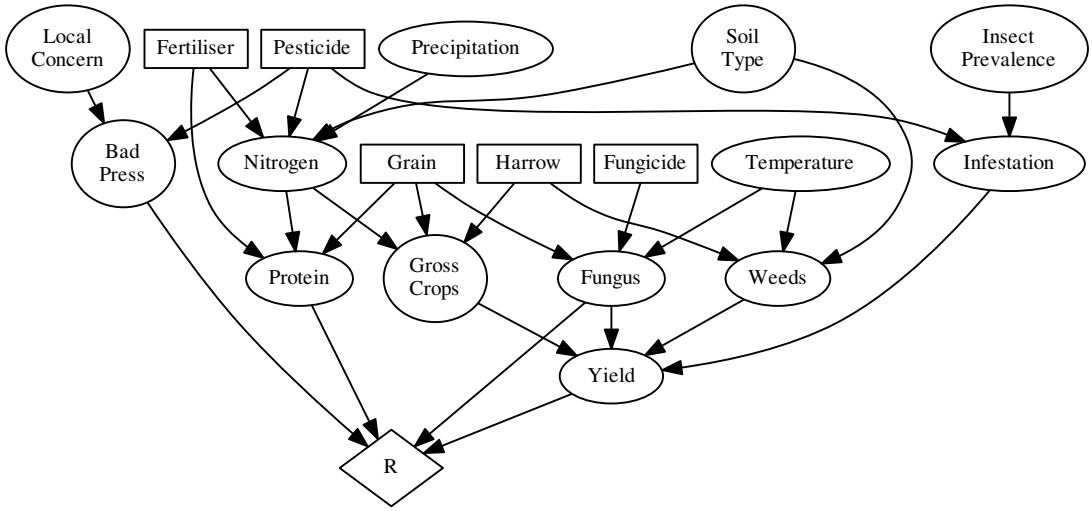


Figure 4.2: *Barley* DN

We also tested agents on three randomly generated DNs of increasing size: 12, 24, and 36 variables. In each, our agent begins with minimal awareness of the true DN ($\mathcal{X}^0 = \{O_1\}, \mathcal{A}^0 = \{A_1\}$, $scope_0(\mathcal{R}) = \{O_1\}$).

In both sets of experiments the agent acts in T trials, using an ε-greedy policy ($\varepsilon = 0.1$). We repeat experiments 50 times and average the results.

We use the *cumulative reward* across trials as our main evaluation metric, which acts as a proxy for the quality of the agent's policy over time. To make the results more readable, we apply a discount of 0.99 at each step, resulting in the metric:

$$R_t^{disc} = r_t + 0.99 * R_{t-1}^{disc} \tag{4.32}$$

We test several variants of our agent to show our approach is effective. The **default** agent follows algorithm 3 as is, with parameters $\kappa = 0.001$, $\tau = 100$, $\rho = 0.1$, $\gamma = 0.99$, $K = 5.0$, $\mu = 10$, $\beta = 0.01$ in equations (3.4), (4.7), (4.8), (3.8), and (3.10). The **nonConservative** agent does not conserve information about $\mathcal{P}$, Pa, nor θ via (3.8) or (3.10) when it discovers a new factor. Instead, it discards all trials and reverts to the original prior of (3.4). We include this agent to show the value of conserving information as $\mathcal{X}$ and $\mathcal{A}$ expand. The **non-relevant** agent is like the default, but does not include any constraints of the form (4.17) when searching for Pa^*. This means the agent might learn DNs where variables are completely disconnected. The **truePolicy** and **random** agents start knowing the true DN, and execute an ε-greedy version of π_+, or choose a random action respectively. These agents give an upper/lower bound

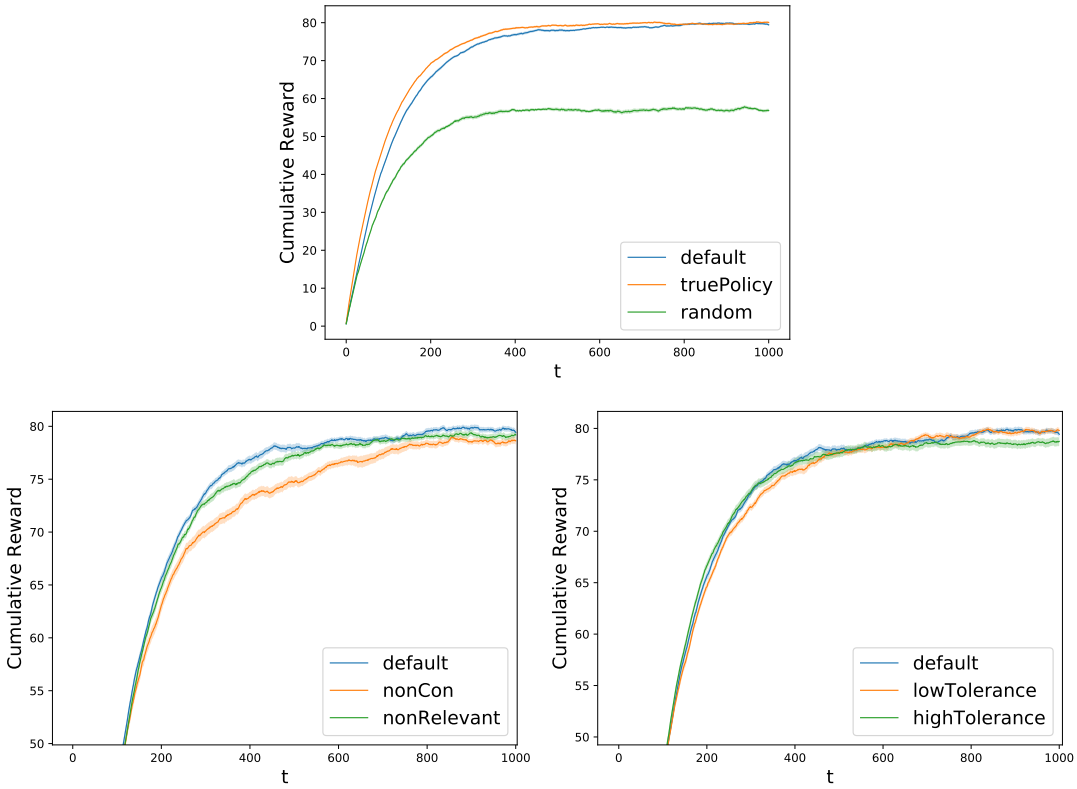


Figure 4.3: Rewards of each agent on *Barley* DN ($T = 1000$). Shaded areas represent standard error from the mean.

on performance. The **lowTolerance** and **highTolerance** agents change the expert's tolerance to $\beta = 0.001$ and $\beta = 0.1$. On the largest DN (36 variables), we also run a few additional tests: the **uniform-prior** agent replaces the cost of additional parents in equation (3.4) to $\rho = 0.5$, while the **no-pruning** agent replaces the threshold for reasonable parents in the Buntine lattice update to $\kappa = 0.0$. These tests were included to verify the benefits of the principle of *minimality* (as briefly discussed in Section 1.3 of the introduction). In other words, the notion that favouring simpler models over complex ones should result in faster learning times without harming the quality of the learned DN.

4.4.1 Results and Discussion

Across all tasks, our default agent converges to the optimal policy, despite starting unaware of factors critical to success. Figures 4.3, 4.5, 4.6 and 4.7 show the cumulative (discounted) rewards gathered by the default agent compared to the non-conservative,

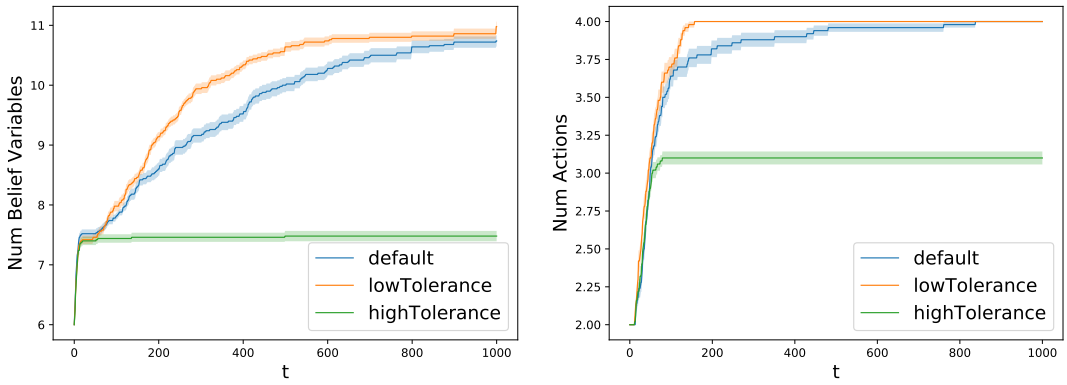


Figure 4.4: Size of $\mathcal{X}$ and $\mathcal{A}$ on *Barley* DN

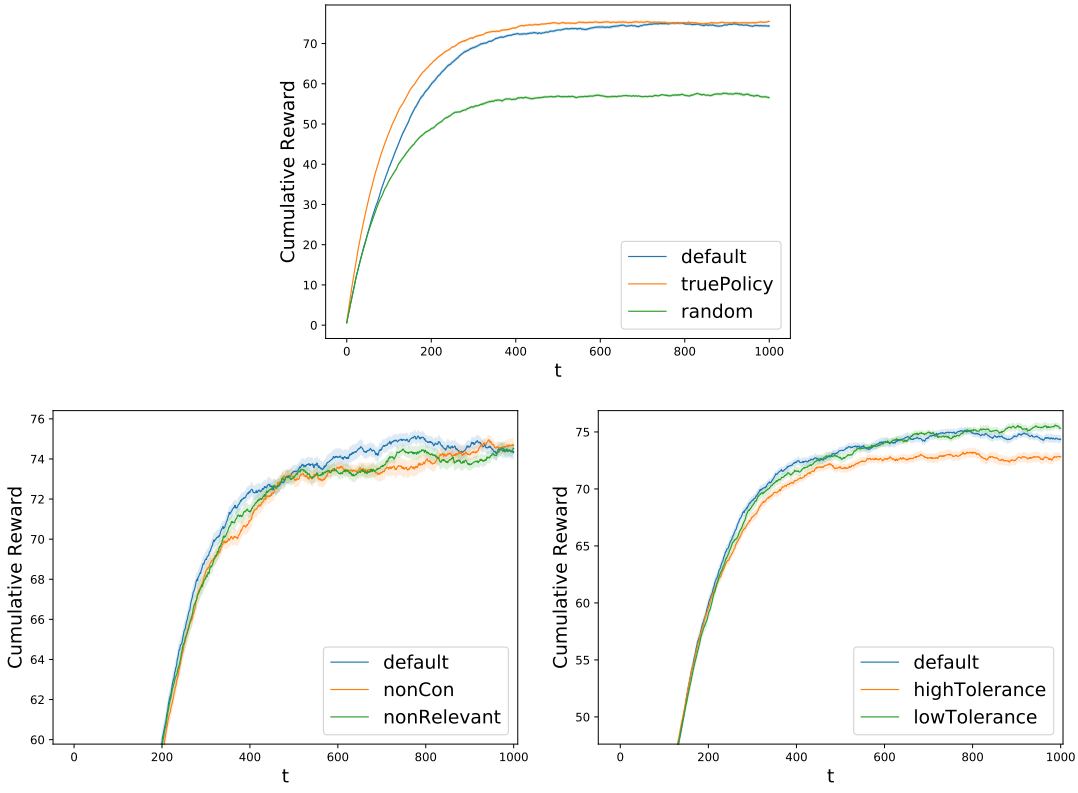


Figure 4.5: Cumulative rewards for Small DN (12 variables, $T = 1000$ trials)

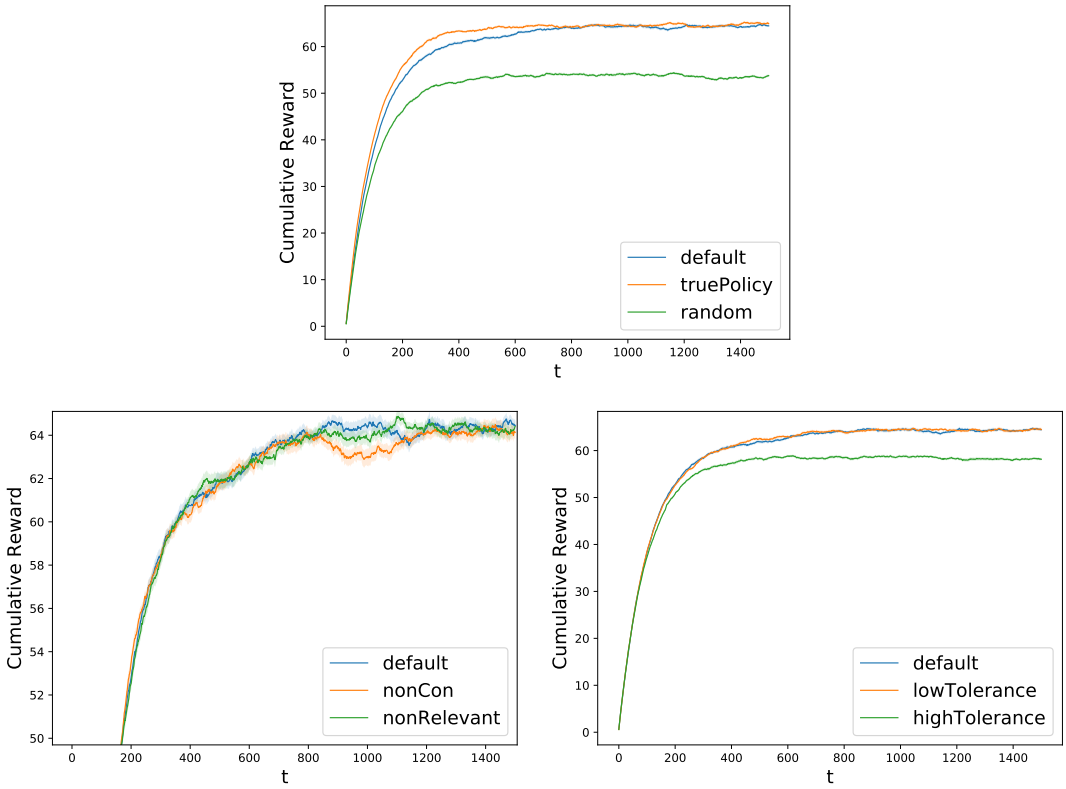


Figure 4.6: Cumulative rewards for Medium DN (24 variables, $T = 1500$ trials)

non-relevant, true-policy, high-tolerance, and low-tolerance agents. The default agent converges to the optimal policy, and does so faster than its non-conservative counterpart. This difference is most pronounced in the large DN task, where the agent is required to expand its awareness to a greater number of unforeseen actions and beliefs to converge on the optimal policy. In the smaller tasks, learning the probabilistic dependencies between the smaller number of variables can be done with a relatively small number of trials, so the ability to conserve previous information is much less important.

The difference in reward compared to the *non-relevant* agent is smaller than expected. We suspect this is because actions which exert a strong causal influence over $scope_+(\mathcal{R})$ will be an ancestor to $scope_+(\mathcal{R})$ in the most likely probabilistic structure (Pa^*), regardless of whether we enforce (4.17). Figure 4.10 illustrates this point with a typical example DN structure learned by each respective agent. Actions the non-relevant agent leaves disconnected are those which exert little causal influence over $scope_+(\mathcal{R})$ (and thus are less likely to have a strong effect on the optimal policy). This means that how much abandoning connectivity affects performance depends on how noisy the causation in the domain model is, which we don't know with certainty in advance.

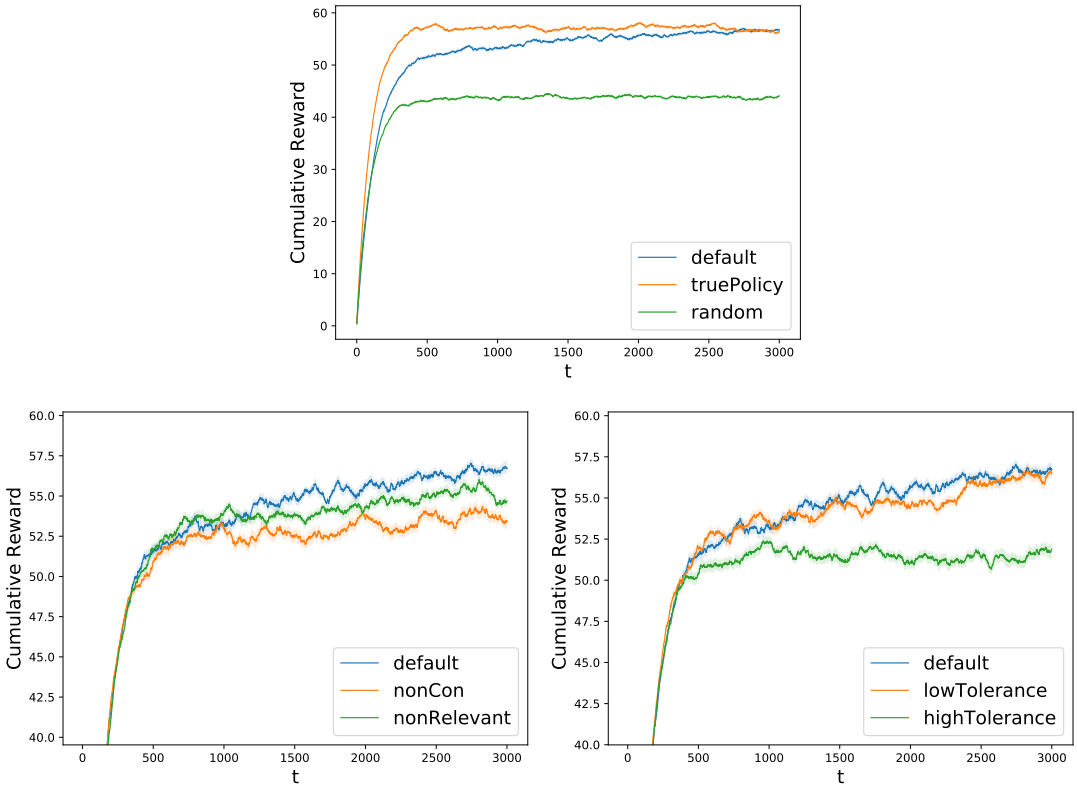
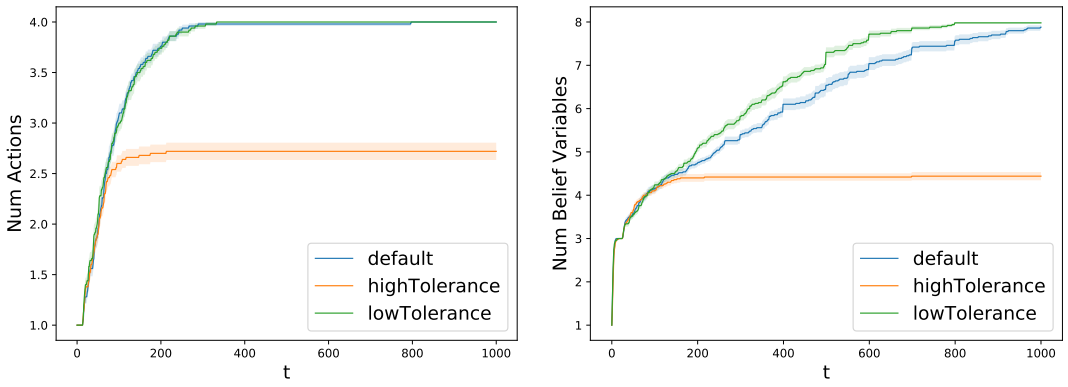


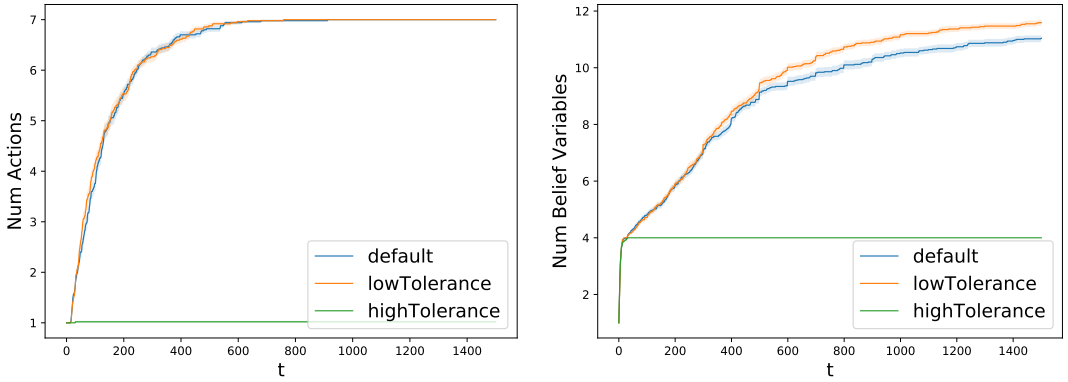
Figure 4.7: Cumulative rewards for Large DN (36 variables, $T = 3000$ trials)

Figures 4.3 to 4.7 also show how the agent's performance differs when varying the expert's tolerance level: with a high tolerance, the agent takes longer to converge towards a good policy and also tends to learn a (marginally) worse final policy. Figures 4.4 and 4.8 give a clue as to why: Early on, the agent's policy becomes "good enough" for the high tolerance expert, meaning the expert does not reveal the last few extra action variables or beliefs which might help the agent gain small increases in its total accumulated reward.

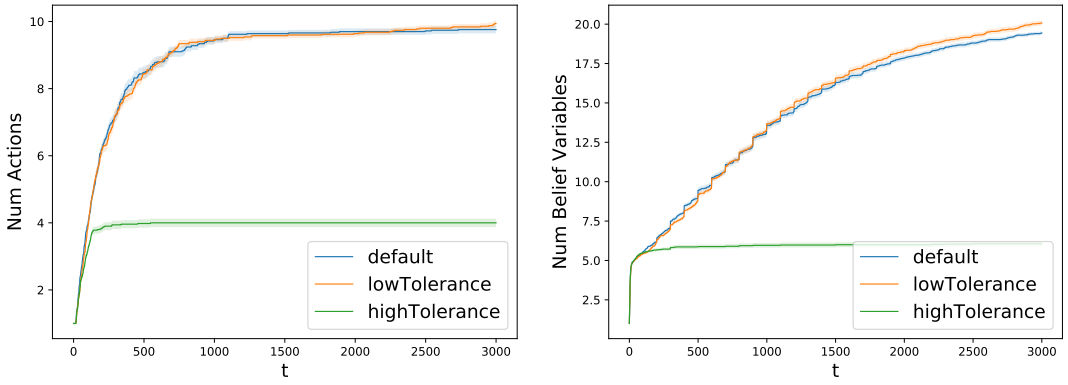
We also measured the total run-time of each agent across each task to see if this revealed any further insights. As Figure 4.9 shows, we found that the non-conservative agent tended to take longer to compute an updated DN in each time step as compared to the other agents, despite becoming aware of similar numbers of belief and action variables. The reason for this is that, by conserving information about the relative probabilities of parent set structures between discoveries, the default agent is able to more aggressively prune the set of reasonable parent sets $\mathcal{P}$ at each time step. In contrast, the non-conservative agent resets its structural priors upon each new discovery, meaning the set of reasonable parent sets tends to remain much larger for a longer period of time.



(a) Small DN

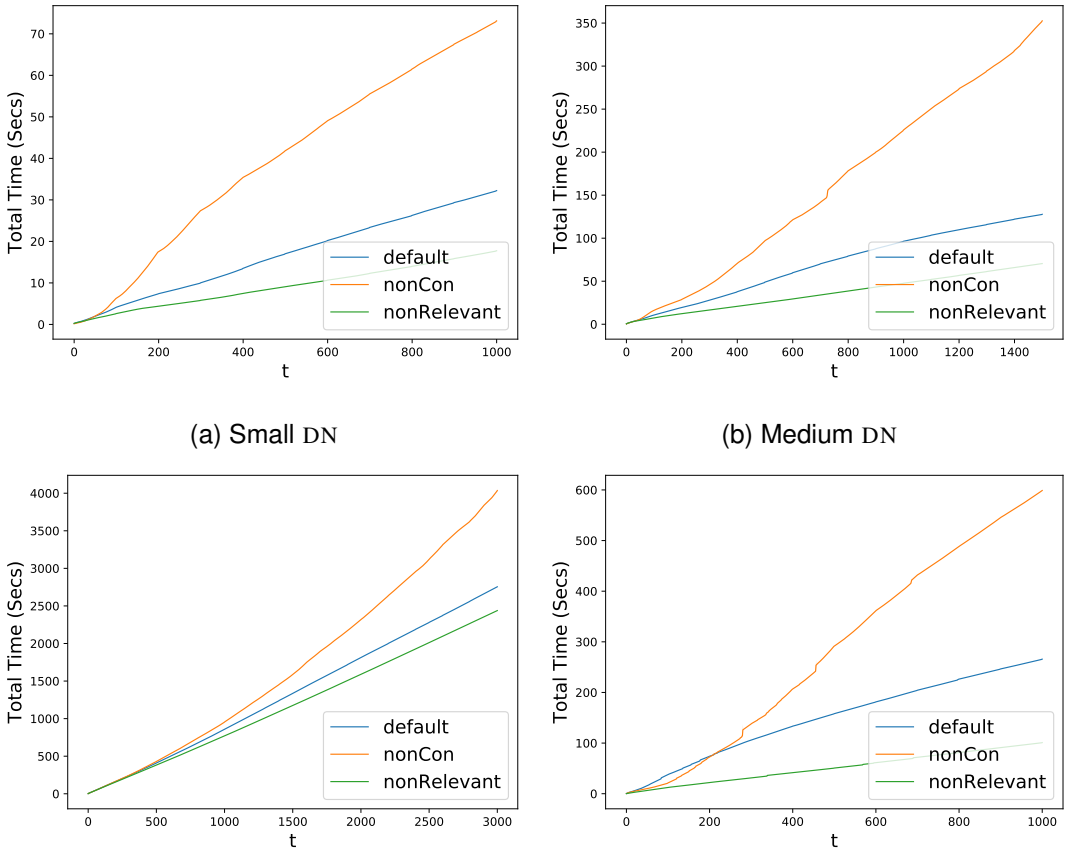


(b) Medium DN



(c) Large DN

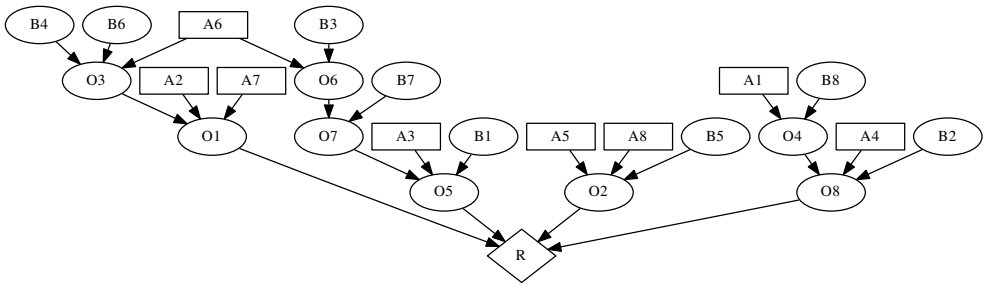
Figure 4.8: Size of $\mathcal{X}$ and $\mathcal{A}$



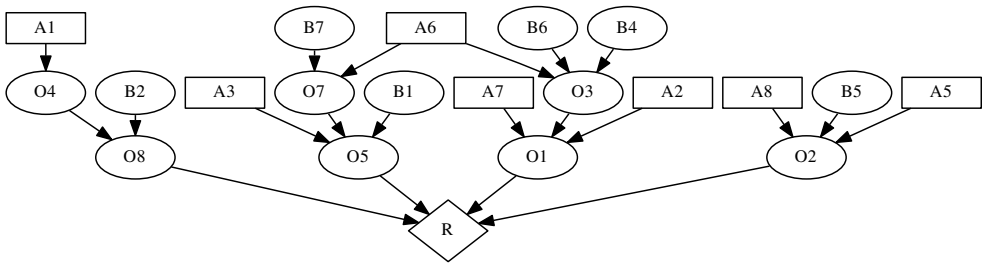
(a) Small DN (b) Medium DN

(c) Large DN (d) Barley DN

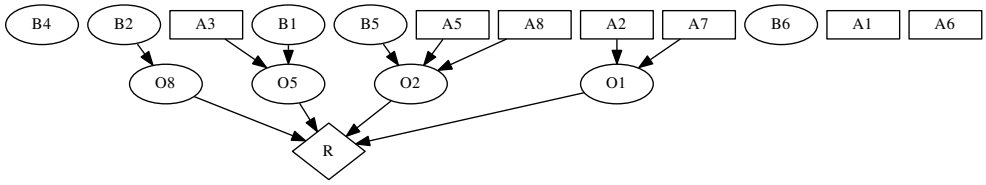
Figure 4.9: Total Run-time of each agent in seconds



(a) True DN



(b) *Default* agent



(c) *Non-relevant* agent

Figure 4.10: Examples of DN structures learned by agents on the *medium* DN task

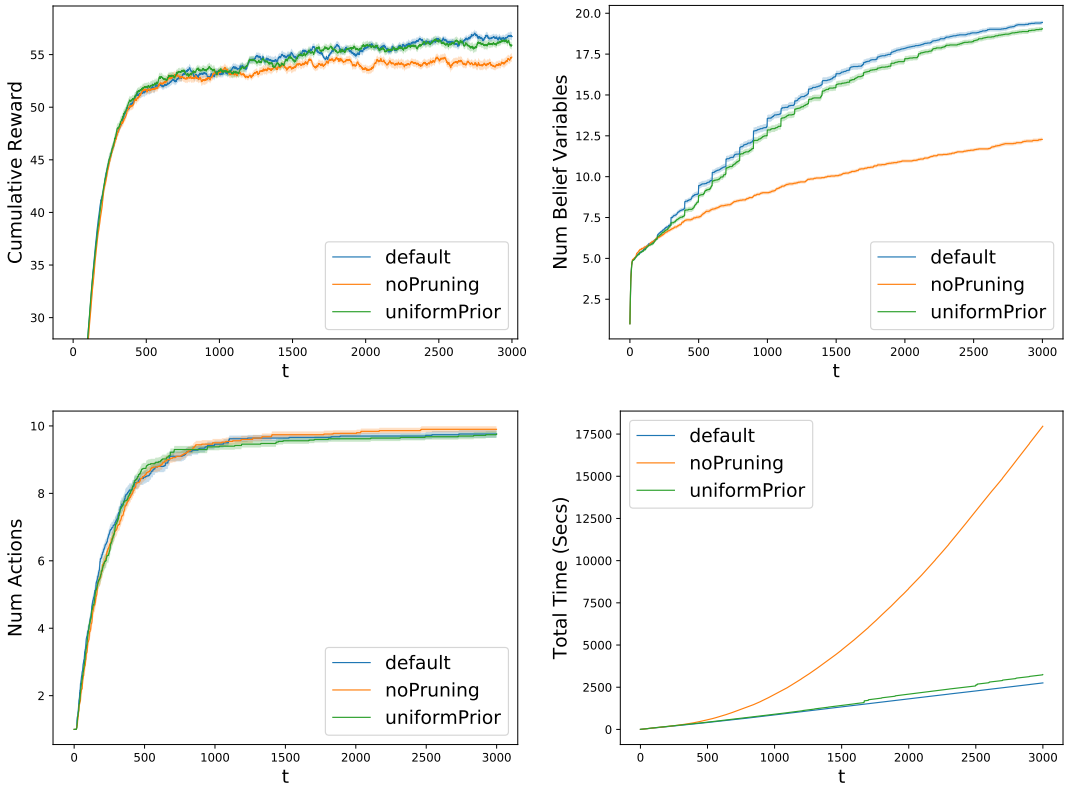
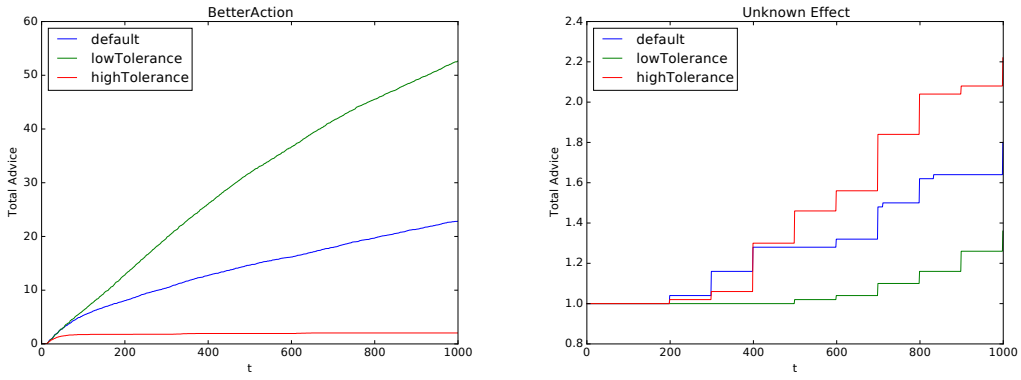


Figure 4.11: Rewards, number of variables, and time taken of agents on the large DN task, with varying levels of minimality. The shaded areas represent the standard error from the mean.



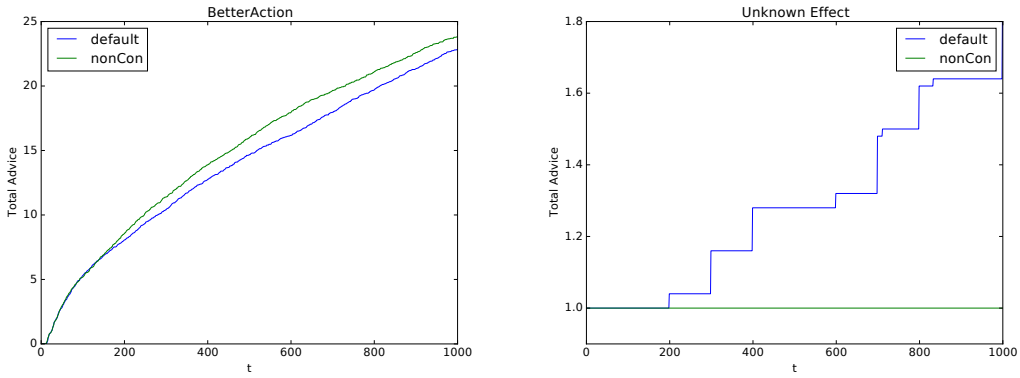
(a) Better Action Advice (Eq 4.12) (b) Unknown Effect (Eq 4.28)

Figure 4.12: Average number of messages sent by the experts of varying tolerances on the *Barley* DN.

Figure 4.11 shows the results comparing the rewards, variables learned, and time taken of the default agent compared to the no-pruning and uniform-prior agents. As expected, the effect of not pruning the set of reasonable parents is that the run-time is orders of magnitude longer than for the default agent. In fact, the no-pruning agent is vastly slower despite typically learning a smaller number of belief variables throughout learning.[6] As for the uniform-prior agent, we found only a marginal difference in run-time compared to the default agent. This is largely because the initial structural prior $P(Pa)$ tends to only make a difference in the initial time steps when data is sparse. Further, despite setting the structural prior to be uniform, there is still an implicit sense of minimality encoded in the structural posterior (equation (3.1)) itself: since we integrate over all possible parameter settings, structures with fewer parameters tend to be a better fit to small amounts of data than complex structures with many parameters. These experiments therefore verify our initial intuition that assuming minimality would result in faster run-times without harming the quality of the models learned by the agent.

Figures 4.12 and 4.13 show the number of messages sent from the expert to the agent on the Barley task, and when each message was sent. The trends shown for these figures apply across the other DNs as well. For the default setup, the expert offers a correction in less than 2% of trials, and reveals only 1-2 edges of the true task's structure over the course of the experiment. Our system therefore succeeds in its goal to avoid incessant

[6]The reason the no-pruning agent tends to learn a smaller number of belief variables is because, if all parent-sets are considered reasonable, then the agent never asks questions of the form (4.27), which cuts off a potential source of new belief variables.



(a) Better Action Advice (Eq 4.12)

(b) Unknown Effect (Eq 4.28)

Figure 4.13: Average number of messages sent to Conservative versus Non-Conservative agents on the *Barley* DN.

interruptions. Most of these corrections arrive in the early trials, then gradually decline in frequency as the agent learns a policy which gets closer to the true optimal policy. Note however, that even when the agent has learned the optimal policy, the expert still occasionally interrupts. This is because the agent is ε-greedy, and so will occasionally take several random (non-optimal) actions consecutively. We could remedy this by having the agent announce its intentions in advance, as done in Torrey and Taylor (2013) (for example, the agent could announce *"I am exploring now!"* whenever it is about to take a random action). However, doing so would restrict the generality of agent and expert's interaction strategies.

As expected, the *lowTolerance* expert interrupts the most, while the *highTolerance* expert interrupts the least. What is more interesting is that the *nonConservative* agent receives more corrections than the *conservative* agent, but ends up gathering less cumulative reward, discovering less vocabulary, and being informed of less edges in the true DN. This is because the non-conservative agent resets its beliefs whenever it discovers a new action or variable, and so receives corrections for areas of the state space that it previously already learned, but "forgot". The conservative agent receives less advice, but the advice more frequently applies to previously unexplored areas of the state space.

4.5 Conclusion

This chapter has shown that through a mixture of domain trials and dialogue evidence, an agent can learn *all* components of a decision problem. This includes not only the structure *Pa* and parameters θ, but also the reward function $\mathcal{R}$, its domain $scope(\mathcal{R})$, and the hypothesis space $\mathcal{A} \cup \mathcal{X}$ itself. Our experiments show that the agent is able to converge on optimal behaviour despite starting with an incomplete hypothesis space, and that the design decisions built on the principles of conserving information as new discoveries are made and minimizing complexity help keep the problem tractable while guiding the agent to better solutions faster.

In this chapter, we dealt only with single-stage problems, where all actions are taken in a single step and the final outcomes and rewards are received immediately. In the next chapter, we look at *sequential* decision problems, where the problem might take place over a large number of interdependent steps, and where rewards and transition probabilities are dependent on the history of the agent's actions.

Chapter 5

Unforeseen Possibilities in Sequential Decision Problems

In the previous chapters, we dealt with "single-stage" decision problems, where the outcome of an agent's current action was independent of previous states or actions. However, many real-world decision problems take place over a sequence of related steps— actions in the past may affect which outcomes (and hence which rewards), are reachable in the future. The contribution of this chapter is to extend both our agent's learning model and the expert's communication framework to support *sequential decision problems*, and once again show that our agent is guaranteed to converge on an optimal policy in both theory (via formal proofs) and practice (via experiments).

5.1 The Learning Task

We focus on learning *complex, episodic, finite state* sequential decision problems by learning *Factored Markov Decision Processes* (FMDPs). We begin with the formalisms for learning optimal behaviour in FMDPs where the agent is fully aware of all possible states and actions, but doesn't know anything about dependencies among those factors (let alone conditional probabilities) nor rewards. We then extend the task to one where the agent starts unaware of relevant variables and actions, and show how the agent overcomes this unawareness with expert aid.

5.1.1 Episodic Markov Decision Processes

An FMDP is a tuple $\langle \mathcal{S}, \mathcal{S}_s, \mathcal{S}_e, A, \mathcal{T}, \mathcal{R} \rangle$, where $\mathcal{S}$ and A are (as before), the set of states and actions;[1] $\mathcal{S}_s \subseteq \mathcal{S}$ are possible starting states of an episode and $\mathcal{S}_e \subseteq \mathcal{S}$ are the end (terminal) states; $\mathcal{T} : \mathcal{S} \times A \times \mathcal{S} \to [0,1]$ is the *markovian transition function* $P(s'|s,a)$ and $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is the immediate reward function.[2] A policy $\pi : \mathcal{S} \times A \to [0,1]$ defining the probability $\pi(s,a)$ that the agent will take action a in state s.

When referring to the local time m in episode n, we denote the current state and reward by $s_{m,n}$ and $r_{m,n} = \mathcal{R}(s_{m,n})$. When referring to the *global time* t across episodes, we denote them by s_t and r_t. For episode n, the *discounted return* is:

$$G^n = \sum_{i=0}^{T} \gamma^i * r_{i,n} \tag{5.1}$$

where $0 \leq \gamma \leq 1$ is the *discount factor* governing how strongly the agent prefers immediate rewards to those further in the future.

The agent's goal is to learn the true *optimal policy* π_+, which maximizes not just the agent's immediate reward, but the expected discounted return across all states. The *value function* $V_\pi(s)$ defines the expected return when following a given policy π, while the related *action-value function* $Q_\pi(s,a)$ gives the expected return of taking action a in state s, and thereafter following π:

$$V_\pi(s) = \mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,\pi(s)) V_\pi(s') \tag{5.2}$$

$$Q_\pi(s,a) = \mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V_\pi(s') \tag{5.3}$$

When $\mathcal{T}$ and $\mathcal{R}$ are known, π_+ can be calculated via value iteration (Sutton and Barto, 1998). Further, as we did for DNs, we can measure the expected loss in discounted return of following a policy π versus π_+ using (5.4), which we refer to as the *policy error*. If the agent's policy is unknown, we can approximate the policy error using (5.5):

[1] In contrast to previous chapters, we do not factor actions as assignments to a set of variables.

[2] Other works allow $\mathcal{R}$ to depend on the action and/or resulting state (i.e., $\mathcal{R} : \mathcal{S} \times A \times \mathcal{S} \to \mathbb{R}$), but we ignore this detail here.

$$Err(\pi) = \sum_{s_0 \in \mathcal{S}_s} P(s_0) \left(V_{\pi_+}(s_0) - V_{\pi}(s_0) \right) \tag{5.4}$$

$$Err(t, t+k) = \left(\sum_{s_0 \in \mathcal{S}_s} P(s_0) V_{\pi_+}(s) \right) - \frac{\sum_{i=t}^{t+k} G^i}{k} \tag{5.5}$$

If all episodes eventually terminate, then (5.5) will converge to (5.4). If we assume as before that our agent is using an ε-greedy policy ($\varepsilon > 0$), then termination in almost all episodic FMDPs is guaranteed:

Definition 1 (Proper Policy)**.** *A policy π is proper if, from all states $s \in \mathcal{S}$, acting according to π guarantees one eventually reaches some terminal state $s' \in \mathcal{S}_e$.*

Theorem 3. *If a proper policy π exists for a given episodic MDP, then any policy which is ε-greedy with respect to A is also proper*

5.2 Learning FMDPs when fully aware

If $\mathcal{T}$ and/or $\mathcal{R}$ are unknown, then the agent must estimate their values using the data $D_{0:t} = [d_0, \ldots, d_t]$ gathered from its interaction with the environment. At time t, the *sequential trial* $d_t = \langle s_t, a_t, s_{t+1}, r_{t+1} \rangle$ represents the current state s_t, the action a_t, the resulting state s_{t+1} and the reward r_{t+1} given on entering s_{t+1}. As with DNs, FMDPs allow one to learn the transition structure of large MDPs by representing states as a joint assignment to a set of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. That is, $s \in v(\mathcal{X})$. Similarly, the reward function is defined as a function $\mathcal{R} : v(scope(\mathcal{R})) \to \mathbb{R}$, where $scope(\mathcal{R}) \subseteq \mathcal{X}$ are variables which uniquely determine the reward received in each state. To exploit conditional independence, $\mathcal{T}$ is then represented as a separate *dynamic bayesian network* (DBN) (Dean and Kanazawa, 1989) for each action. That is, $\mathcal{T} = \{dbn_{a_1}, \ldots, dbn_{a_n}\}$, where $dbn_a = \langle Pa^a, \theta_a \rangle$. Here, Pa^a is a DAG with nodes $\{X_1, \ldots, X_n, X'_1, \ldots, X'_n\}$ where, as is standard, node X_i denotes the value of variable $X_i \in \mathcal{X}$ at the current time step, while X'_i denotes the value of the same variable at the next time step. For each X'_i, $Pa^a_{X'_i}$ defines the parents of X'_i. These are the set of variables on which the value of X'_i depends. It is assumed that X'_i is independent of all other variables. We also make the common assumption that our DBNs contain no *synchronic arcs* Degris and Sigaud (2010):

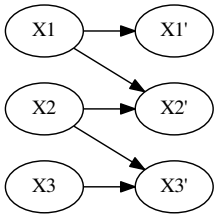


Figure 5.1: Example 3-variable DBN. An arc from X_i to X'_j means that X_i's value in the previous time step influences X_j's value in the current one

$$\forall i, \forall a, Pa^a_{X'_i} \subseteq \{X_1, \ldots, X_n\} \tag{5.6}$$

This structure, along with the associated parameters θ^a, allow us to write transition probabilities as a product of independent factors:

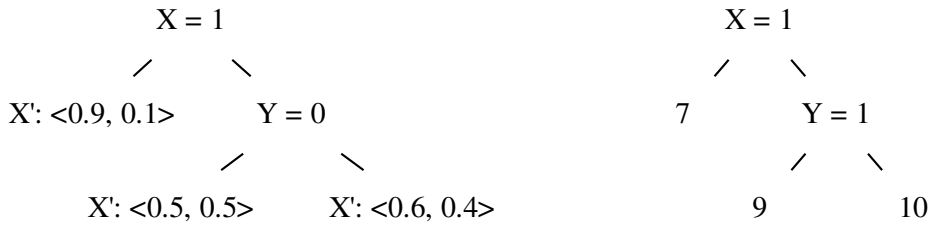
$$P(s'|s,a) = \prod_{X \in \mathcal{X}} \theta^a_{s'[X], s[Pa^a_{X'}]} \tag{5.7}$$

Where $\theta^a_{X'=i, Pa^a_{X'}=j}$ denotes the probability of variable X taking on value i given that the agent performed action a when the variables $Pa^a_{X'}$ had assignment j in the current time step.[3] Figure 5.1 show an example of a simple 3-variable DBN for a given action a. Notice, for example, that X_2's value in the current time step is dependent only on the values of X_1 and X_2 in the previous time-step.

Exploiting independence among belief variables does not necessarily result in a compact representation of V_π, since even those variables which are not part of $scope(\mathcal{R})$ tend to become *entangled* and exert some influence on expected returns as the sequential problem progresses. We must therefore also exploit the *context-specific independencies* between assignments by representing both $\mathcal{T}$ and $\mathcal{R}$ as *decision trees*, rather than just as tables of values.

Figures 5.2a and 5.2b show an example decision tree for a reward function and conditional probability distribution. The leaves are either immediate rewards, or a distribution over the values of a variable. The non-leaves are *test nodes*, which perform a binary test of the form $(X = i?)$ to check whether the variable X takes on the value i in the current state. Notice in 5.2a that when $X = 1$ is true, the distribution over X' is conditionally independent of Y.

[3] When the context is clear, we condense this notation to $\theta^a_{i,j}$.



(a) CPT for $P(X'|Pa^a_{X'})$, where $Pa^a_{X'} = \{Y, Z\}$ (b) Reward tree where $scope(\mathcal{R}) = \{X, Y\}$

Figure 5.2: Examples of Decision Trees

Given trials $D_{0:t}$ we can estimate the most likely DBN structure, decision tree structures, and parameters, then subsequently use these estimates to estimate V_{π_+} via a series of steps.

5.2.1 Estimating DBN structure

We can find the most likely DBN structure Pa^a* using the *BD-score*, as we did in previous chapters:

$$P(Pa^a|D_{0:t}) = \prod_{X \in \mathcal{X}} BD_t(X', Pa^a_{X'}) \tag{5.8}$$

$$BD_t(X', Pa^a_{X'}) = P(Pa^a_{X'}) \prod_{j \in v(Pa^a_{X'})} \frac{\beta(N^a_{1,j} + \alpha^a_{1,j}, \ldots, N^a_{m,j} + \alpha^a_{m,j})}{\beta(\alpha^a_{1,j}, \ldots, \alpha^a_{m,j})} \tag{5.9}$$

$$\tag{5.10}$$

Here, $N^a_{X'=i, Pa^a_{X'}=j}$ denotes the number of trials in $D_{0:t}$ where the agent took action a in a state where variables $Pa^a_{X'}$ had the joint assignment j, resulting in a state where X takes on the value i.

In most sequential decision problems, one usually assumes that each of the agent's actions affect only a small number of features in the environment. Further, while some variables are inherently stochastic, it seems sensible to assume that variables which are not acted upon by the agent are likely to maintain their current value from one time step to another (Boutilier and Goldszmidt, 1996). We can encode this assumption into the agent's prior beliefs about Pa using equation (5.11):

$$P(Pa^a_{X'}) = \prod_{Y \in Pa^a_{X'}} f(Y,X) \prod_{Z \notin \Pi_{X'}} (1 - f(Z,X)) \tag{5.11}$$

$$f(Y,X) = \begin{cases} \rho_{same} & \text{if } Y = X \\ \rho_{diff} & \text{otherwise} \end{cases} \tag{5.12}$$

Where the parameter $0.5 < \rho_{same} < 1$ governs how strongly the agent favours making X a parent of X', and parameter $0 < \rho_{diff} \leq 0.5$ determines how averse the agent is to allowing other variables to affect X'. As in equation (3.4), equation (5.11) states that the probability of parent set $Pa^a_{X'}$ is then the product of each individual parent's presence or absence.

As before, if the space of possible DBNs is too large, we can restrict the parent sets considered reasonable by restricting the maximum in-degree, or using heuristics such as *sparse candidate* (Friedman *et al.*, 1999), or the using the structure lattice of Buntine (1991). However, notice that because of our assumption that the FMDP does not contain any synchronous arcs, we do not have to worry about whether our final learned structure will be a DAG, and can therefore simplify the task of finding Pa^a* by finding each $Pa^a_{X'}*$ independently.

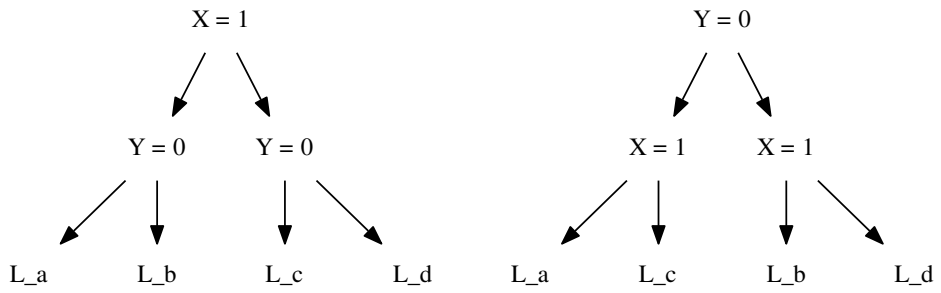
5.2.2 Estimating conditional probability trees

Given each $Pa^a_{X'}$, we can then compute each variable's most likely conditional probability tree structure DT^{a*}_X using equations (5.13-5.14):

$$P(\mathrm{DT}^a_X | Pa^a_{X'}, D_{0:t}) \propto P(\mathrm{DT}^a_X) P(D_{0:t} | \mathrm{DT}^a_X) \tag{5.13}$$

$$P(D_{0:t} | \mathrm{DT}^a_X) = \prod_{\ell \in Leaves(\mathrm{DT}^a_X)} \frac{\beta(N^a_{1|\ell} + \alpha^a_{1|\ell}, \ \ldots \ , N^a_{m|\ell} + \alpha^a_{m|\ell})}{\beta(\alpha^a_{1|\ell}, \ \ldots \ , \alpha^a_{m|\ell})} \tag{5.14}$$

Here, each leaf node $\ell \in Leaves(\mathrm{DT}^a_X)$ is described by its *branch label* within the decision tree (i.e., the set of assignments on the path starting from the root of the tree and ending at ℓ). For example, in Figure 5.2a, the leaf with value $X' : \langle 0.5, 0.5 \rangle$ has branch label $\{X = 0, Y = 0\}$. Since we have assumed our DBN structure is $Pa^a_{X'}$, we only consider tree structures where all test nodes are members of $Pa^a_{X'}$.



(a) Before transposition (b) After transposition

Figure 5.3: Example of transposing root node $X = 1$ with $Y = 0$

Rather than evaluating the probabilities of all possible DT structures each step, we can cache the relevant $N_{i|\ell}$ counts at each node, and incrementally update the most likely DT as new trials arrive using *incremental tree induction* (ITI), as show in Algorithm 4. ITI is described in detail in Utgoff *et al.* (1997), but broadly works as follows:

When a new trial d_t arrives, it starts at the tree's root and passes through the test nodes corresponding to d_t's particular assignments. At each node d_t touches, UPDATE-INFO updates the scores (5.13) for each potential test at that node, then marks that node as being out of date ("stale"), before finally recording d_t at the appropriate leaf node. Updating the test probabilities at each test node may mean that the current test at a particular node is no longer the most likely one. If so, we replace old test with the new most probable one, and recursively restructure (via TRANSPOSE-TREE) the decision tree such that the nodes below this one have valid leaves and data.

Figure 5.3 shows an example DT with leaves L_a, L_b, L_c, and L_d being transposed in response to a new domain trial making $Y = 0$ a more likely test candidate for the root node than $X = 1$. Notice that, for many transposition operations, we can exploit shared nodes further down to avoid recounting domain trials. Full details of the operation of updates and transpositions in ITI can be found in Utgoff *et al.* (1997)

5.2.2.1 Handling changing values of $Pa^a_{X'}$

The above explanation assumed $Pa^a_{X'}$ remained the same between ITI updates. However, if a new trial d_t makes some alternative structure $Pa^{a,t+1}_{X'} \neq Pa^{a,t}_{X'}$ more likely via equation (5.8), then our current DT^a_X may be invalid. If this occurs, there are three scenarios to consider:

Algorithm 4 Incremental Tree Induction (Utgoff *et al.* (1997))

```
function INCREMENTAL-UPDATE(node, example)
    ADD-EXAMPLE-TO-TREE(node, example)
    ENSURE-BEST-TEST(node)

function ADD-EXAMPLE-TO-TREE(node, example)
    if node is a leaf then
        node.examples ← node.examples ∪ example
        if node should be converted to decision node then
            CONVERT-TO-DECISION-NODE(node)
            node.stale ← false
            for all e in node.examples do
                if TEST-PASSED(node, e) then
                    ADD-EXAMPLE-TO-TREE(node.left, e)
                else
                    ADD-EXAMPLE-TO-TREE(node.right, e)
    else
        UPDATE-INFO(node, example)
        node.stale ← true
        if TEST-PASSED(node, e) then
            ADD-EXAMPLE-TO-TREE(node.left, e)
        else
            ADD-EXAMPLE-TO-TREE(node.right, e)

function ENSURE-BEST-TEST(node)
    if node is decision node ∧ node.stale then
        bestTest ← FIND-BEST-TEST(node)
        if bestTest ≠ node.currentTest then
            TRANSPOSE-TREE(node, bestTest)
```

First, if $Pa_{X'}^{a,t+1} \subset Pa_{X'}^{a,t}$, then all the sufficient statistics we need to update $\text{DT}_{X'}^{a}$ are already present in $\text{DT}_{X'}^{a,t}$. As an example, suppose at time t we had $Pa_{X'}^{a,t} = \{X,Y,Z\}$, but in at time $t+1$ we have $Pa_{X'}^{a,t+1} = \{X,Y\}$. In the trivial scenario where Z is not currently being used at any test node in $\text{DT}_{X'}^{a}$, we do not need to make any changes to the structure of $\text{DT}_{X'}^{a}$—we simply remove Z as a candidate test for each node and we are done. If Z *is* being used at some test node, then we must replace it with the next most likely test out of X and Y, triggering a tree re-structuring as outlined in the previous section. Since we already have the relevant counts and probabilities for these tests stored at the relevant node, the information we need to conduct this restructuring is immediately available.

Second, if $Pa_{X'}^{a,t} \subset Pa_{X'}^{a,t+1}$, this may require more computation. Suppose this time we have $Pa_{X'}^{a,t} = \{X\}$ and $Pa_{X'}^{a,t+1} = \{X,Y\}$. Since we do not have the counts for tests on Y immediately available at the previous test nodes, we must compute them. Fortunately, we can do this without a full pass over $D_{0:t}$, as we already have the counts we need from the calculation of $Pa^{a,t+1}$'s likelihood. For example, if we wish to compute (5.14) for a decision node with attribute test "$Y = 0$?", then we can compute the required sufficient statistics (e.g. $N_{X'=0|Y=0}$) from a combination of the statistics used to compute (5.8) and (5.14) (e.g. $N_{X'=0|X=0,Y=0} + N_{X'=0|X=1,Y=0}$).

Third, if neither $Pa_{X'}^{a,t+1} \subset Pa_{X'}^{a,t}$ nor $Pa_{X'}^{a,t} \subset Pa_{X'}^{a,t+1}$ is true, we can concatenate the solutions to the first two scenarios—first we remove all the missing variables, then add in all additional variables.

5.2.3 Estimating the reward tree

We can use ITI to learn a tree structure for $\mathcal{R}$ in the same way, with only a few minor differences. First, we restrict our node test to members of $scope(\mathcal{R})$, rather than $Pa_{X'}^{a}$. Second, rather than recording a *list* of trials at each leaf node, each leaf node ℓ stores a *set* of relevant state/reward pairs SRP_ℓ (since we are not interested in storing the frequencies of duplicate trials):

$$\text{SRP}_\ell = \{\langle s,r\rangle | \lambda(s) = \ell, \langle s',a,s,r\rangle \in D_{0:t}\} \tag{5.15}$$

Here, $\lambda(s)$ is a function which takes a state assignment and returns the leaf to which s would be assigned in our current DT.

The third difference is how the agent decides which test to install at each node. We are no longer learning a probabilistic parameter, so equation (5.13) is unsuitable. Instead, we wish to choose the tests which will minimize the *expected entropy* of the reward received in a given state. Equation (5.16) gives the expected entropy for splitting a leaf node with test sp:

$$-\frac{N_{sp}}{|D_{0:t}|}H(R_{sp}|sp) - \frac{N_{\neg sp}}{|D_{0:t}|}H(R_{sp}|\neg sp) \tag{5.16}$$

$$H(R|sp) = \sum_{r \in R} \frac{N_{r|sp}}{N_{sp}} \log\left(\frac{N_{r|sp}}{N_{sp}}\right) \tag{5.17}$$

Where N_{sp} is the number of trials at the current node which pass test sp, $N_{\neg sp}$ is the number which fail it, and R_{sp} is the set of unique rewards present in states which pass the test sp (formally, $R_{sp} = \{r | \langle s, r\rangle \in srp_\ell \wedge sp(s) == true\}$). $H(R|sp)$ is the conditional entropy of possible reward values R given sp, and $Nr|sp$ is the number of state descriptions which pass the test sp and whose reward is r.

5.2.4 Estimating V_{π_+}

Once our agent has an estimate of $\mathcal{T}$ and $\mathcal{R}$, we can then use *structured value iteration* (SVI) (Boutilier *et al.*, 2000)—a variant of value iteration which works with decision trees instead of tables—to compute a compact representation of V_{π_+}. Algorithm 5 shows an outline of an incremental version of SVI (iSVI) (Degris *et al.*, 2006), which allows the agent to gradually update its beliefs about the optimal value function in response to incoming trials. The algorithm takes the current estimate of the reward and transition functions ($\mathcal{R}_t$ and $\mathcal{T}_t$), along with the previous estimate of the optimal value function (V_{t-1}), and combines them to produce a new estimate for each state-action function (Q_t^a), and value function (V_t).

Algorithm 5 Incremental SVI (Degris *et al.*, 2006)

1: **function** INCSVI($\mathcal{R}_t$, $\mathcal{T}_t$, V_{t-1})
2: $\quad\forall a \in A : Q_t^a \leftarrow \text{REGRESS}(V_{t-1}, dbn_a, \mathcal{R}_t)$
3: $\quad V_t \leftarrow \text{MERGE}(\{Q_t^a : \forall a \in A\})$ (using maximization as the combination function)
4: $\quad$ **return** $\{V_t, \{\forall a \in A : Q_t^a\}\}$

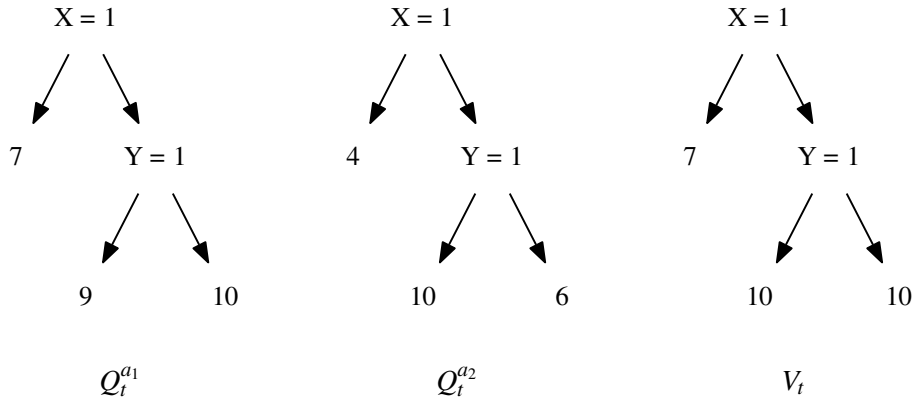


Figure 5.4: Example of merging $Q_t^{a_1}$ and $Q_t^{a_2}$ to form V_t

The first operator REGRESS$(Tree(V_{t-1}), a)$, takes the value-tree from the previous iteration, and produces a tree-structured Q-function Q_t^a for each possible action a:

$$Q_t^a(s) = \sum_{s'} P(s'|s,a)\left(\mathcal{R}(s) + \gamma V_{t-1}(s')\right) \tag{5.18}$$

The second operator MERGE$(\{Q_t^{a_1}, Q_t^{a_2}, \ldots\ Q_t^{a_n},\})$ combines each of the Q-functions at the tth iteration into a single value tree via maximisation. This then represents the value function V_t over all possible states. Figure 5.4 shows an example of Q-trees $Q_t^{a_1}$ and $Q_t^{a_2}$ being merged to produce the latest value function V_t. As with the original value iteration algorithm, SVI is guaranteed to converge to the true optimal policy, but unlike regular value iteration, it (in general) produces a more compact description of the final policy function in substantially less computation time. For full details how the MERGE and REGRESS function in SVI operate, see Boutilier *et al.* (2000).

This section took an *encapsulated* approach to learning $\mathcal{T}$ (In contrast to a *unified* one in e.g., Degris *et al.* (2006)). This means we separate the task of finding an optimal DBN structure from the task of learning each local DT structure. Such an approach significantly reduces the space of DTs that must be considered, but more importantly, provides us with posterior *distributions* $P(Pa_{X'}^a|D_{0:t})$ over parent structures. We will use these posterior distributions in Section 5.3.2 to conserve information when adapting to new discoveries.

5.3 Overcoming Unawareness

So far, we've assumed our agent was aware of all relevant belief variables in $\mathcal{X}$, all actions A, and all members of $scope(\mathcal{R})$. As in previous chapters, we now drop this assumption. From here onward we denote the true set of belief variables, actions, and reward scope as $\mathcal{X}^+$, A^+ and $scope_+(\mathcal{R})$, and the learner's awareness of them at t as $\mathcal{X}^t$, A^t, and $scope_t(\mathcal{R})$.

The next sections aims to answer two main questions. First, how does the protocol for communication between the agent and expert differ in the case of sequential problems when compared with the interaction we modelled in Chapter 4 for single-shot decision problems? Second, when an agent discovers a new belief variable or action, how can they integrate it into their current FMDP model while conserving what they have learned from past experience?

5.3.1 Expert Guidance

As in the previous chapter, our agent can expand its awareness via advice from an expert, who is once again cooperative, infallible, and has full knowledge of the true FMDP. The three types of advice we identify in this section—advice on a better action, resolving a misunderstanding, and explaining an unexpected reward—are broadly similar to the ones defined in Chapter 3, and their combination guarantees our agent behaves optimally in the long run regardless of initial unawareness. However, the conditions under which the expert utters these types of advice, and the inferences the agent can draw from them differ slightly due to the sequential nature of the task.

5.3.1.1 Better Action Advice

If the expert sees the agent perform a sub-optimal action, it can tell the agent a better action it could have taken instead. For example: "When it is raining, take your umbrella instead of your sun hat". Again, our goal is to avoid incessantly interrupting the agent each time it makes a mistake, so we specify the following conditions for when the agent is performing sufficiently poorly to warrant correction: Let t be the current (global) time step corresponding to the mth step in the nth episode. Similarly, let t', m', n' be the

time the expert last uttered advice. When (5.19-5.21) hold, the expert will utter advice of the form (5.22):

$$t - t' > \mu_{min} \tag{5.19}$$

$$Err(n', n) > \beta \vee m > \mu_{max} \tag{5.20}$$

$$\exists a' \in A^+, Q_{\pi_+}(a', s_{m,n}) > Q_{\pi_+}(a_{m,n}, s_{m,n}) \tag{5.21}$$

$$Q_{\pi_+}(w^s_{m,n}, a') > Q_{\pi_+}(w^s_{m,n}, a_{m,n}) \tag{5.22}$$

Equation (5.19) ensures some minimum time μ_{min} has passed since the expert last gave advice (compare this with equation (4.7)). Equation (5.20) ensures the expert won't interrupt unless its estimate of the agent's policy error is above some threshold β (cf equation (4.8)), or if the agent is unable to reach a terminal state after some reasonable bound μ_{max} (which is required because the agent's unawareness of A^+ may mean its current ε-greedy policy is not proper). If episode n is unfinished, the expert estimates the expected return via the heuristic $G^n \approx \sum_{i=0}^{m-1} \gamma^i r_{i,n} + \gamma^m V_{\pi_+}(s_{m,n})$, i.e., we optimistically assume the agent will follow π_+ from now on. Taken together, μ_{min}, μ_{max} and β describe the expert's *tolerance* towards the agent's mistakes. Finally, (5.21) ensures a better action a' actually exists this step.

Equation (5.22) is the expert's utterance. In a similar fashion to the previous chapter, $w^s_{m,n}$ is a sense ambiguous term, whose intended meaning is the true state $s_{m,n}$ but whose default interpretation by the agent at time t is $s_{m,n}[\mathcal{X}^t]$. The agent can thus interpret this advice in two ways: The first is as a *partial description* of the true problem, which is *monotonically* true regardless of what it learns in future. On hearing (5.22), the agent adds (5.23-5.24) to its list of partial descriptions of the true FMDP:

$$a' \in A^+ \tag{5.23}$$

$$\exists s, s[\mathcal{X}^t] = s_{m,n}[\mathcal{X}^t] \wedge Q_{\pi_+}(s, a') > Q_{\pi_+}(s, a_{m,n}) \tag{5.24}$$

Notice that this differs from the partial description (4.15) from the previous chapter, in that be can the agent can no longer infer that the expert's advice entails the existence of a state with a higher immediate reward. This is because, in a sequential problem, an action may be better in the current state not because it leads to a better expected reward

immediately, but because it increases the probability of entering a higher reward state some time in the future.

Additionally, the agent can choose to add its current *default interpretation* of the advice to its accumulated knowledge:

$$\forall s, s[\mathcal{X}^t] = s_t[\mathcal{X}^t] \implies Q_{\pi_+}(s[\mathcal{X}^t], a') > Q_{\pi_+}(s[\mathcal{X}^t], a) \tag{5.25}$$

The agent can then act on the expert's advice directly by choosing a' whenever $s[\mathcal{X}^t] = s_{m,n}[\mathcal{X}^t]$, regardless of what seems likely from $D_{0:t}$.

Lemma 4 guarantees the expert's dialogue strategy for offering unsolicited advice eventually reveals unforeseen actions to the agent so long as its performance in trials exceeds the expert's tolerance.

Lemma 4. *Consider an* FMDP *where π_+ is proper, an agent with awareness $\mathcal{X}^t \subseteq \mathcal{X}^+, A^t \subset A^+$, and expert acting with respect to (5.19-5.22). If $\exists a \in image(\pi_+), a \notin A^t$ then as $k \to \infty$, either $Err(t, t+k) \to c$ with $c \leq \beta$ or the expert utters (5.21) such that $a' \notin A^t$.*

Outline Proof. If $Err(t, t+k) \to c \leq \beta$, we are done. If not, we must consider two cases—one where an ε-greedy policy over A^t is proper, and one where it is not. If it is, (5.20) will eventually be satisfied, since the expert will gather enough samples to approximate the agent's policy error (which is above β). Further, we know $\exists s, \pi_+(s) = a' \wedge a' \notin A^t$, and that s is reachable by the agent, so (5.21) will eventually be satisfied. If the agent's policy is *not* proper, the agent may visit some set of states $\mathcal{S}' \subset \mathcal{S} \setminus \mathcal{S}_e$ infinitely often (thus satisfying (5.20) for finite μ_{max}). Since π_+ is proper, there must exist some $s \in \mathcal{S}'$ such that $\exists a' \notin A^t, \forall a \in A^t, Q_{\pi_+}(s, a') > Q_{\pi_+}(s, a)$, thus satisfying (5.21). In either case, (5.19) will eventually be satisfied for finite μ_{min}. Since (5.19-5.21) are not mutually exclusive, all three will eventually be true simultaneously, causing the expert to utter (5.22). □

5.3.1.2 Resolving Misunderstandings

We noted before that, just like in the interaction for learning a DN in Chapter 4, the agent's defeasible interpretation of expert advice could result in misunderstandings

about the intended meaning. To illustrate, suppose the agent receives advice (5.26) and (5.27) at times $t-k$ and t:

$$Q_{\pi_+}(w^s_{t-k}, a) > Q_{\pi_+}(w^s_{t-k}, a') \tag{5.26}$$

$$Q_{\pi_+}(w^s_t, a) < Q_{\pi_+}(w^s_t, a') \tag{5.27}$$

While the intended meaning of each statement is true, the agent's default interpretations of w^s_{t-k} and w^s_t may be identical. That is, $s_{t-k}[\mathcal{X}^t] = s_t[\mathcal{X}^t]$. From the agent's perspective, (5.26) and (5.27) conflict, and thus give the agent a clue that its current awareness of $\mathcal{X}^+$ is deficient. To resolve this conflict, the agent asks (5.28) (in words, *"which X has distinct values in s_{t-k} and s_t?"*) and receives an answer of the form (5.29):

$$?\lambda X(X \in \mathcal{X}^+ \wedge s_{t-k}[X] \neq s_t[X]) \tag{5.28}$$

$$X \in \mathcal{X}^+ \tag{5.29}$$

Notice there may be multiple variables in $\mathcal{X}^+ \setminus \mathcal{X}^t$ whose assignments differ in s_{t-k} and s_t. Thus, the expert's answer can be *non-exhaustive*, providing the minimum amount of information to resolve the agent's conflict without necessarily explaining all components of the task. This means the agent must abandon its previous defeasible interpretation of (5.25), but can keep (5.23-5.24), as these are true regardless of known variables. Lemma 5 guarantees the expert will (eventually) reveal new belief variables, provided that the agent's unawareness is such that misunderstandings are still possible.

Lemma 5. *Consider an* FMDP *where π_+ is proper and an agent with awareness $\mathcal{X}^t \subset \mathcal{X}^+, image(\pi_+) \subseteq A^t \subseteq A^+$. If $\exists s \exists s' \neq s, s[\mathcal{X}^t] = s'[\mathcal{X}^t]$, and $\pi_+(s) \neq \pi_+(s')$, then as $k \to \infty$, either $Err(t, t+k) \to c$ $(c \leq \beta)$, or the expert utters (5.29) such that $X \notin \mathcal{X}^t$*

Outline Proof. The ε-greedy agent is aware of all actions in $image(\pi_+)$, so can visit all states reachable via π_+, including s and s'. If $Err(t, t+k) \to c, c \leq \beta$, we are done. If not, then at some time $t+k_1$ the agent will do a' in s, causing the expert to utter $Q(w^s_{t+k_1}, a) > Q(w^s_{t+k_1}, a')$. Similarly, at some time $t+k_2$, the agent will receive advice $Q(w^s_{t+k_2}, a') > Q(w^s_{t+k_2}, a)$. Since $s_{t+k_1}[\mathcal{X}^t] = s_{t+k_2}[\mathcal{X}^t]$, the two pieces of advice appear to the agent to conflict, so the agent will ask (5.28) with answer $X \notin \mathcal{X}^t$. □

5.3.1.3 Unexpected Rewards

In typical FMDPs (where the agent is assumed fully aware of $\mathcal{X}^+, A^+$, and $scope_+(\mathcal{R})$), we tend only to think of the trials as providing *counts*, but for an unaware agent, a trial $d_t = \langle s_t, a_t, s_{t+1}, r_{t+1} \rangle$ also encodes *monotonic information*:

$$\exists s, s[\mathcal{X}^t] = s_{t+1} \wedge \mathcal{R}_+(s) = r_{t+1} \tag{5.30}$$

This constrains the form of $\mathcal{R}$ the agent must learn. Recall that $scope_t(\mathcal{R})$, may be only a subset $scope_+(\mathcal{R})$, so it might be impossible to construct an $\mathcal{R} : v(scope_t(\mathcal{R})) \to \mathbb{R}$ satisfying all descriptions (5.30) gathered so far. Further, those extra variables in $scope_+(\mathcal{R}) \setminus scope_t(\mathcal{R})$ may not be in $\mathcal{X}^t$. To resolve this, if the agent fails to construct a valid reward function, it asks (5.31) (in words, *"which variable X (that I don't already know) is in scope*$(\mathcal{R})$*?"*), receiving an answer (5.32):

$$?\lambda X (X \in scope_+(\mathcal{R}) \bigwedge_{X' \in scope_t(\mathcal{R})} X \neq X') \tag{5.31}$$

$$X \in scope_+(\mathcal{R}) \wedge X \in \mathcal{X}^+ \tag{5.32}$$

Again, the agent may be unaware of many variables in $scope_+(\mathcal{R})$, so (5.32) may be *non exhaustive*. Even so, we can guarantee that the agent's learned reward function eventually equals $\mathcal{R}_+$:

Lemma 6. *Consider an* FMDP *where* π_+ *is proper and an agent with awareness* $A^t \subseteq A^+$, $\mathcal{X}^t \subseteq \mathcal{X}^+$, $scope_t(\mathcal{R}) \subseteq scope_+(\mathcal{R})$. *As* $k \to \infty$, *there exists a K such that for all* $k \geq K$, $\mathcal{R}_{t+k}(s) = \mathcal{R}_+(s)$ *for all states s reachable using* A^t.

Outline Proof. If s is reachable using A^t, then as $k \to \infty$, an ε-greedy agent will eventually enter s at some time i, receive reward $\mathcal{R}_+(s)$, and update its current reward function so that $\mathcal{R}_i(s) = \mathcal{R}_+(s)$. If the agent has previously encountered another s' such that $s[scope_t(\mathcal{R})] = s'[scope_t(\mathcal{R})]$ and $\mathcal{R}_+(s) \neq \mathcal{R}_+(s')$, the partial descriptions (5.30) for s and s' will conflict. The agent resolves this by asking (5.31), receiving an answer differentiating s from s' in $\mathcal{R}_+$. □

5.3.2 Adapting the Transition Function

Section 5.3.1 showed three ways the agent could expand its awareness of $\mathcal{X}$, A, and $scope(\mathcal{R})$. If we wish to improve on the naive approach of restarting learning when faced with such expansions, we must now specify how the agent adapts $\mathcal{T}$ and $\mathcal{R}$ to such discoveries.

Adapting $\mathcal{T}$ upon discovering a new action a' at time t is simple: Since the agent hasn't performed a' in any previous trial, it can just create a new DBN, $dbn_{a'}$, using the priors outlined in Section 5.2.1. Our new model at time t then becomes $\mathcal{T} = \{dbn^t_{a_1}, \ldots dbn^t_{a_n}\} \cup \{dbn_{a'}\}$.

The more difficult issue is adapting $\mathcal{T}$ upon discovering a new belief variable Z. The main problem is that the agent's current distributions over DBNs no longer cover all possible parent sets for each variable, nor all DTs. For example, the current distribution over $Pa^a_{X'}$ does not include the possibility that Z is a parent of X'. Worse, since we assume in general that the agent *cannot observe Z's past values* in $D_{0:t}$, it cannot observe the true value of $N^a_{Z=i|j}$, nor $N^a_{X=i|Pa^a_{X'}=j}$ when $Z \in Pa^a_{X'}$. The α-parameters involving Z are also undefined, yet we need them to calculate structure probabilities (5.9, 5.13) and parameters via (3.6).

As done for DNs in the last chapter, our method discards the data gathered during the learner's previous deficient view of the hypothesis space, but *conserves* the relative *posterior* probabilities learned from past data to construct new *priors* for the Pa^a, DT^a and θ^a in the expanded belief space.

5.3.2.1 Parent Set Priors

On discovering Z, the agent must update $P(Pa^a_{X'})$ for each $X \neq Z$ and $a \in A^t$ to include parent sets containing Z. In (5.33) we construct a new prior $P'(Pa^a_{X'})$ using the old posterior (where C is a normalizing constant):

$$P'(Pa^a_{X'}) = \begin{cases} (1-\rho)P(Pa^a_{X'}|D_{0:t}) & \text{if } Z \notin Pa^a_{X'} \\ \rho P((Pa^a_{X'} \setminus Z)|D_{0:t}) & \text{otherwise} \end{cases} \tag{5.33}$$

This preserves the likelihoods among the parent sets that do not include Z. It also maintains our bias towards simpler structures by re-assigning only a portion ρ of the

probability mass to parent sets including Z. To define $P(Pa^a_{Z'})$—the distribution over parent sets for the newly discovered variable Z—we default to (5.11), since the agent has no evidence (yet) concerning Z's parents.

5.3.2.2 Decision Tree and Parameter Priors

We must also update $P(\mathrm{DT}^a_X|Pa^a_{X'})$, and $P(\theta^a_{X,Pa^a_{X'}}|Pa^a_{X'})$ to accommodate Z. Here, we return to the issue of the counts $N^a_{i|j}$ and the associated α-parameters. As mentioned earlier, we wish to avoid the complexity of estimating Z's past values. Instead, we *throw away* the past counts of $N^a_{i|j}$, but *retain* the relative likelihoods they gave rise to by packing these into new α-parameters, as shown in (5.34-5.35):

$$\alpha^a_{X=i|Y=j} := \begin{cases} \frac{K}{|v(Z\cup Y)|} & \text{if } X = Z \\ \frac{K}{|v(Y)|} P(i, j[Y \setminus Z]|dbn^t_a) & \text{else} \end{cases} \tag{5.34}$$

$$P'(\mathrm{DT}^a_{X'}|Pa^a_{X'}) \propto \prod_{\ell \in leaves(\mathrm{DT}^a_{X'})} \beta(\alpha^a_{X'=1|\ell}, \ldots, \alpha^a_{X'=n|\ell}) \tag{5.35}$$

Equation (5.34) summarizes $D_{0:t}$ via inferences on the old best DBNs, then encodes these inferences in the new α-parameters. The revised α-parameters ensure the new tree structure prior and expected parameters defined via (5.35) and (3.6) bias towards models the agent previously thought were likely. Indeed, the larger the (user-specified) K parameter is, the more the distributions learned *before* discovering Z influence the agent's reasoning *after* discovering Z.

5.3.3 Adapting the Reward and Value Functions

On becoming aware that a new Z is part of $scope_+(\mathcal{R})$, the agent may wish to restructure its reward tree. This is because awareness that $Z \in scope_+(\mathcal{R})$ means there are tests of the form $Z = i$ that the agent has not yet tried which may produce a more compact tree. In the language of ITI, the current test nodes are "stale", and must be re-checked to see if a replacement test would yield a tree with better information gain.

The only added complication here is that, since the agent may have previously unaware of Z (i.e., $Z \notin \mathcal{X}^t$), we know that the previous set of state-reward descriptions at each node given by equation (5.15) were actually only *partial* state-reward descriptions:

$$\text{SRP}_\ell = \{\langle s[\mathcal{X}^t], r\rangle | \lambda(s[\mathcal{X}^t]) = \ell, \langle s'[\mathcal{X}^t], a, s[\mathcal{X}^t], r\rangle \in D_{0:t}\} \tag{5.36}$$

We can still test on the values of partial state-reward descriptions by following the ITI convention that any partial description where Z is missing automatically fails any test on the value of Z. Further, if some future trial d_j has a strictly more precise partial description than some previous trial d_i ($i < j$), then can completely replace the partial state-reward description of d_i by d_j.

Once we have updated $\mathcal{T}$ and $\mathcal{R}$, there is no need to make further changes to V_t in response to a new action a' or variable Z. In effect, this encodes our conservative intuition that the true V_{π_+} is more likely to be closer to the agent's current estimate V_t than some arbitrary value function. The agent essentially assumes (in absence of further information) that the value of a state is indifferent to this newly discovered factor. In subsequent trials where the agent performs a' or observes Z, algorithm 5 ensures information about this new factor is incorporated into the agent's value function.

Algorithm 6 outlines how the agent updates $\mathcal{T}$, $\mathcal{R}$, and V in response to new data and expert advice. Given algorithm 6, Theorem 4 guarantees our agent behaves indistinguishably from a near-optimal policy in the long run (where by "near-optimal" we mean that it is within the bounds β of the expert's tolerance to the agent's mistakes), regardless of initial awareness (provided all $X \in \mathcal{X}^+$ are relevant to expressing the optimal policy).

Theorem 4. *Consider an* FMDP *where π_+ is proper and an agent with initial awareness $\mathcal{X}^0 \subseteq \mathcal{X}^+, A^0 \subseteq A^+$, and $scope_0(\mathcal{R}) \subseteq scope_+(\mathcal{R})$ acts according to algorithm 6. If for all $X \in \mathcal{X}^+$, there exists a pair of states s, s' such that $s[\mathcal{X}^+ \setminus X] = s'[\mathcal{X}^+ \setminus X]$, $s[X] \neq s'[X]$, and $\pi_+(s) \neq \pi_+(s')$, then as $t \to \infty$, $Err(0,t) \to c$ such that $c \leq \beta$*

Outline Proof. By repeatedly applying Theorems 4 - 6, we have that if $Err(0,t)$ has not yet converged to $c \leq \beta$, then there exists a K where $image(\pi_+) \subseteq A^K$, $\mathcal{X}^K = \mathcal{X}^+$, and $\mathcal{R}_K(s) = \mathcal{R}_+(s)$ for all s reachable with A^K. Thus $\mathcal{X}^K$, A^K, $\mathcal{R}_K$ define a separate MDP for which the agent is fully aware, but has the same π_+ as the original. All episodes terminate, so the agent's estimate of $\mathcal{T}$ eventually approximates the true transition function, V_t converges to the V_{π_+}, and thus $Err(0,t) \to 0$. □

Algorithm 6 Learning FMDPs with Unawareness

function LEARNFMDPU(A^0, $\mathcal{X}^0$, $\mathcal{T}_0$, Q_0, V_0, s_0)
  for $t = 1 \ldots maxTrials$ **do**
    $\langle s_t, r_t \rangle \leftarrow \varepsilon$-GREEDY($s_{t-1}$, Q_{t-1}, $adv_{0:t-1}$)
    $\langle \mathcal{T}_t, \mathcal{R}_t \rangle \leftarrow$ Add $\langle s_t, r_t \rangle$ via (5.9-3.6) & ITI
    if Update to $\mathcal{R}_t$ fails **then**
      $Z \leftarrow$ Ask expert (5.28)
      $\langle scope_t(\mathcal{R}), \mathcal{X}^t \rangle \leftarrow$ Append Z to each
      $\mathcal{R}_t \leftarrow$ Update via ITI
    if (5.19-5.21) are true **then**
      $adv_t \leftarrow$ Expert advice of form (5.22)
      if adv_t mentions action $a' \notin A^{t-1}$ **then**
        $\mathcal{A}^t \leftarrow \mathcal{A}^{t-1} \cup \{a'\}$
        $\mathcal{T}_t \leftarrow \mathcal{T}_{t-1} \cup \{dbn_{a'}\}$ made via (5.11)
      if $adv_{0:t-1}$ conflicts with adv_t **then**
        $Z \leftarrow$ Ask expert (5.28)
        $\mathcal{X}^t \leftarrow \mathcal{X}^{t-1} \cup \{Z\}$
    if $\mathcal{X}^{t-1} \neq \mathcal{X}^t$ **then**
      $\mathcal{T}_t \leftarrow$ Update via (5.34, 5.9, 5.35, 5.13, 3.6)
    $\langle V_t, Q_t \rangle \leftarrow$ INCSVI($\mathcal{R}_t, \mathcal{T}_t, \mathcal{V}_{t-1}$)

5.4 Experiments

Our experiments show that agents following algorithm 6 converge to near-optimal behaviour in both theory and practice. Further, we show that conserving information on $\mathcal{T}$ and V gathered before each new discovery allows our agent learn faster than one which abandons this information. We do not investigate assigning an explicit budget to agent-expert communication, leaving this to future work. However we do show how varying the expert's tolerance affects the agent's performance.

We test agents on two well-known problems: *Coffee-Robot* and *Factory*.[4] In each, our agent begins with only partial awareness of $\mathcal{X}^+$, A^+ and $scope_+(\mathcal{R})$. The agent takes actions for T time steps, using an ε-greedy policy ($\varepsilon = 0.1$). When the agent enters a terminal state, we reset it to one of the initial states randomly. We use the *cumulative reward* across all trials as our evaluation metric, which acts as a proxy for the quality of the agent's policy over time. To make the results more readable, we apply a discount of 0.99 at each step, resulting in the metric $R_t^{disc} = r_t + 0.99 * R_{t-1}^{disc}$.

We test several variants of our agent to show the effectiveness of our approach. The **default** agent follows algorithm 6 as is, with parameters $\rho = 0.1$, $K = 5.0$, $\mu_{min} = 10$, $\beta = 0.1$, $\mu_{max} = 50$ in equations (5.11), (5.33), (5.34), and (5.19-5.21) respectively. The **nonConservative** agent does not conserve information about V, nor $\mathcal{T}$ via (5.33-5.35) when a new factor is discovered. Instead, it resets V and $\mathcal{T}$ to their initial values. This agent is included to show the value of conserving past information as $\mathcal{X}$ and A expand. The **truePolicy** and **random** agents start with full knowledge of the true FMDP, and execute an ε-greedy version of π_+, or a choose random action respectively. These agents provide an upper/lower bound on performance. The **lowTolerance / highTolerance** agents change the expert's tolerance to $\beta = 0.01$ and $\beta = 0.5$.

5.4.1 Coffee Robot

Coffee-Robot is a small sequential problem where a robot must purchase coffee from a cafe, then return it to their owner. Also, the robot gets wet if it has no umbrella when it rains. The problem has 6 boolean variables—HUC (user has coffee), HRC (robot has coffee), R (raining), W (wet), L (location), U (umbrella)— and 4 actions—MOVE, DELC, BUYC and GETU— making 256 state/action pairs. The terminal states are those

[4] Full specifications at `https://cs.uwaterloo.ca/~jhoey/research/spudd/index.php`

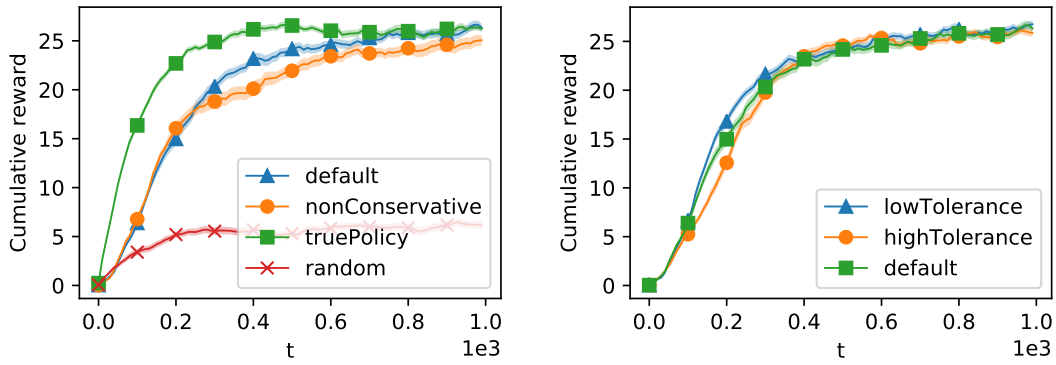


Figure 5.5: Cumulative Rewards on *Coffee Robot* task, $T = 1000$, averaged over 50 experiments. Shaded areas represent the standard error from the mean.

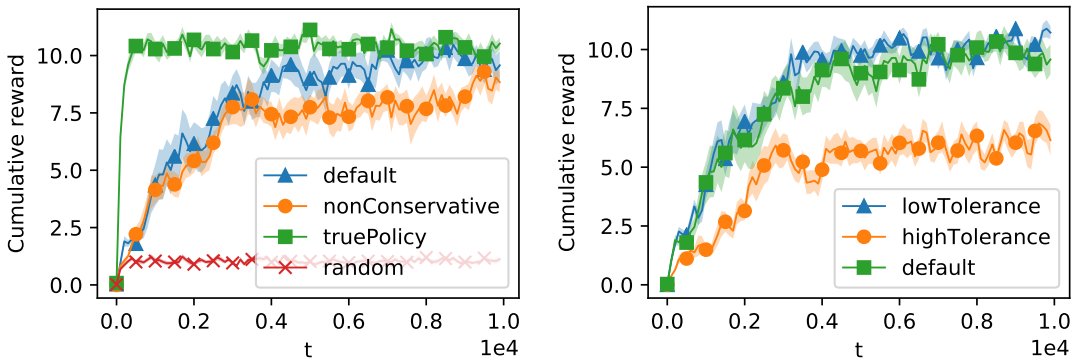
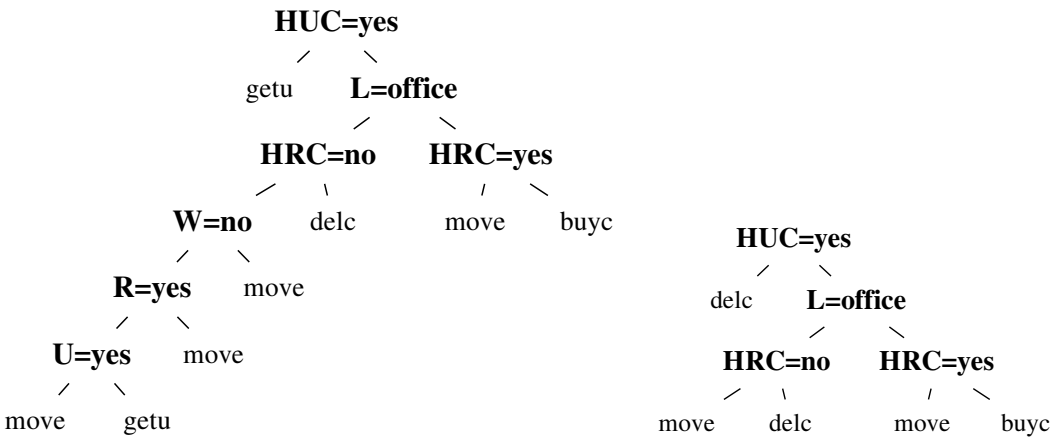


Figure 5.6: Cumulative Rewards on *Factory* task, $T = 10000$, averaged over 20 experiments. Shaded areas represent the standard error from the mean.

where HUC $= 1$; initial states are all non-terminal ones. Our agent has initial awareness $A^0 = \{\text{MOVE}\}$, $\mathcal{X}^0 = scope_0(\mathcal{R}) = \{\text{HUC}\}$ and discount factor $\gamma = 0.8$.[5]

Figure 5.5 shows each agent's (discounted) cumulative reward. Despite starting unaware of factors critical to success, the default agent quickly discovers the relevant actions and beliefs with the expert's aid, and converges on the optimal policy. The non-conservative agent also learns the optimal policy, but takes longer. This shows the value of conserving $\mathcal{T}$ and V on discovering new beliefs. We also see how expert tolerance affects performance. The agent paired with high tolerance expert learns a (marginally) worse final policy, but this makes little difference to cumulative reward. Figure 5.7 shows why: The agent learned a "good enough" policy, so the expert doesn't reveal the "get umbrella" (GETU) action, which yields only a minor increase in reward. Figure 5.8

[5]Original setting was $\gamma = 0.9$. Changed to make π_+ proper.



(a) Default Tolerance (b) High Tolerance

Figure 5.7: Typical final policy depending on tolerance

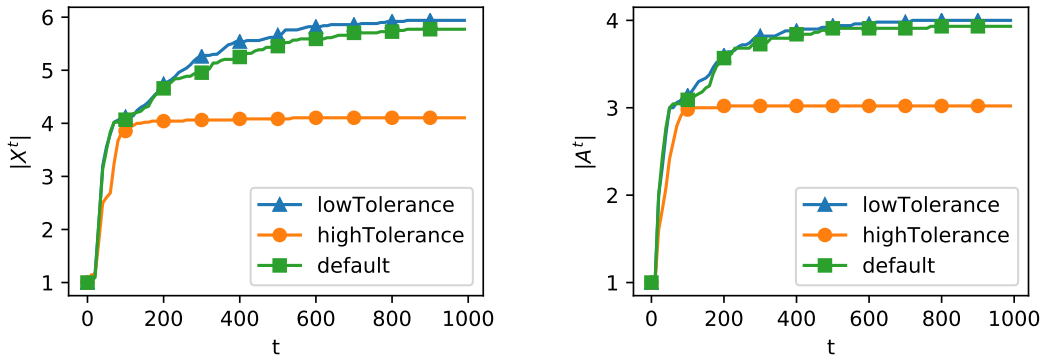
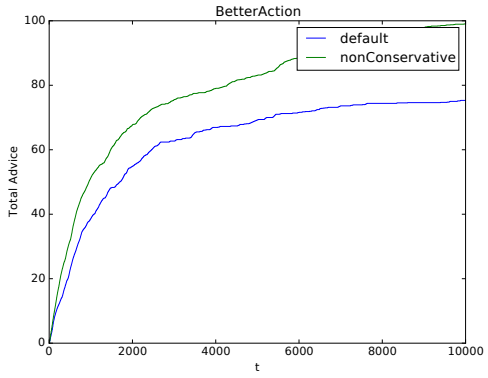


Figure 5.8: Awareness of $|\mathcal{X}^+|$ and $|A^+|$ on *Coffee Robot*

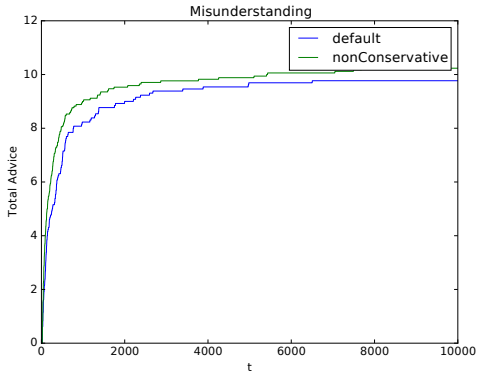
supports this explanation, showing how more tolerant experts reveals less variables over time.

5.4.2 Factory

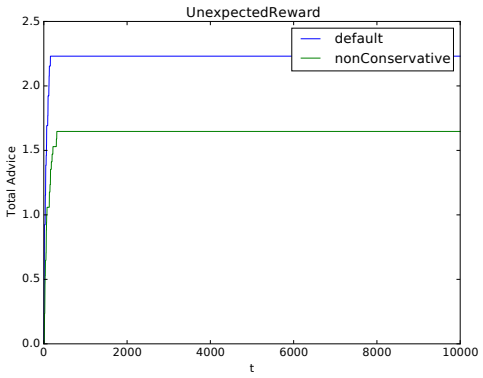
Factory is a larger problem ($|A^+| = 14$, $\mathcal{X}^+ = 14$, 774144 state/action pairs), which shows our method works on more realistically sized tasks. Here, an agent must shape, paint and connect two widgets to create products of varying quality. Some actions (like bolting) produce high quality products, whereas others (like gluing) produce low quality products. The agent receives a higher reward for producing goods which match the de-



(a) Better Action (Eq 5.22)

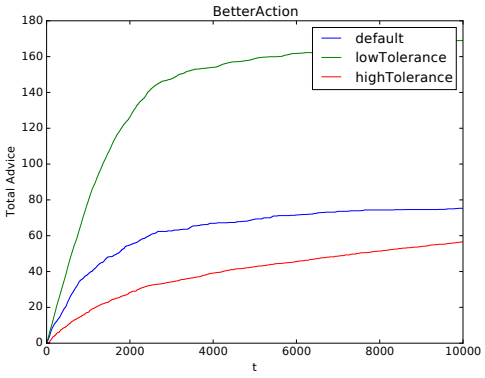


(b) Resolve a Misunderstanding (Eq 5.29)

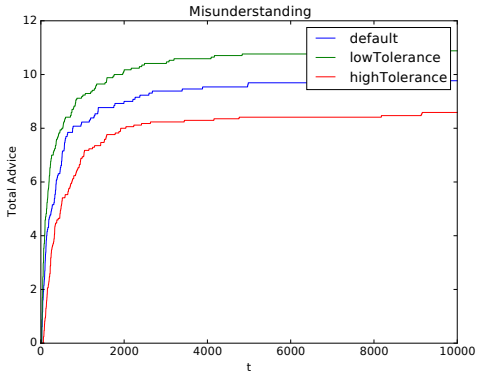


(c) Explaining an unexpected reward (Eq 5.32)

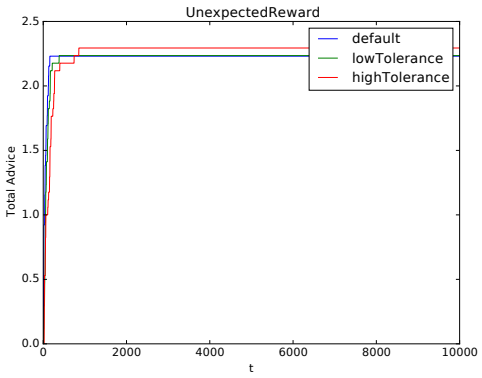
Figure 5.9: Comparing the average number of messages sent by the expert on *Factory* when paired with a Conservative versus Non-Conservative agent.



(a) Better Action (Eq 5.22)



(b) Resolve a Misunderstanding (Eq 5.29)



(c) Explaining an unexpected reward (Eq 5.32)

Figure 5.10: Comparing the average number of messages sent by the experts of varying tolerance on *Factory*.

manded quality.[6]. The terminal states are those where CONNECTED $= 1$; initial states are non-terminals where it is possible to connect two components. Our agent's initial awareness is $\mathcal{X}^0 = scope_0(\mathcal{R}) = \{\text{CONNECTED}\}$, $A^0 = \{\text{BOLT, GLUE, DRILLA, DRILLB}\}$, with $\gamma = 0.9$. This represents a simplified task where the agent thinks the only goal is connecting the widgets.

Figure 5.6 shows results similar to previous experiments. The default agent converges on optimal behaviour, and does so quicker than the non-conservative agent. Varying the expert's tolerance now has a larger effect on the rate at which factors are discovered and on convergence towards the optimal policy (presumably because there are many more unforeseen variables/actions the agent can discover in this larger problem).

Figures 5.9 and 5.10 break down the messages sent between the agent and expert over time. These results mirror the findings from section 4.4: The expert provides advice in only a tiny fraction of trials, agents receive less corrections as the task progresses, tolerant experts interrupt less frequently, and the non-conservative agent performs more poorly despite receiving more corrective advice in total.

5.5 Conclusion

This chapter extended our agent-expert framework to sequential decision problems, and showed that our agent could learn optimal behaviour on both small and large FMDPs, even when starting unaware of factors critical to success. Echoing the results from previous chapters, we have shown that conserving one's beliefs about the transition and value functions improves the effectiveness of learning.

[6]Rewards were scaled to range 0.0-1.0 and, to make π_+ proper, terminal states which previously gave 0 reward were given a small reward of 0.01.

Chapter 6

Conclusion / Future Work

In this thesis, we have proposed a model-based learning framework for dealing with both single-stage and multi-step sequential decision tasks. However, unlike traditional approaches, our learning agent begins *unaware* of the true actions and belief variables which are relevant to the problem. Our model exploits two kinds of evidence to discover and learn to exploit factors that the agent starts out unaware of: as is traditional, the agent uses evidence from trial and error; and in addition, the learner uses evidence from an extended conversation that it has with a domain expert, where the expert's utterances are contextually relevant to the agent's latest move: either a dialogue move where the agent has asked a pertinent question to resolve its current dilemma in inference, which the expert then answers, or unsolicited advice about better options in a context where the agent has been performing sufficiently poorly on the domain level task. We have confined the conversation to these types of moves because our eventual aim is for our model to support learning optimal behaviours from human teachers in a scenario that is similar to one where an apprentice interacts with a human teacher.

We showed that our formal protocol for communication between the learner and expert, while confined to the types of dialogue moves that are natural in a human setting (although the language is not natural, the type of dialogue acts are!), we were able to provably guarantee that the agent would eventually converge on optimal behaviour, despite starting unaware of factors that may be critical to success.

By conducting a range of experiments across various bayesian networks, decision networks and factored markov decision processes, we showed that our agent model converged upon optimal behaviour not just in theory, but also in practice. Further, the

experiments justified our intuitions that an agent which conserves information about its previously learned model upon discovering new unforeseen factors would learn faster than one which abandoned such information.

This work is takes some of the necessary initial steps in devising an autonomous agent which can learn in an ever-expanding hypothesis space, and as a result, there are many fruitful areas in which this work could be extended:

Expanding the Capabilities of the Expert: To restrict the scope of this work, we placed a number of strong assumptions on the expert and its communication with the learning agent. However, our ultimate long term goal is to extended such work to have a *human* expert assistant. In this case, many of the assumptions that we have made in this work would have to be lifted. For example, we assume the expert never makes a mistake, and has full knowledge of the true underlying decision task. Most human experts do make mistakes, or have only partial knowledge of a problem. Indeed, it is because humans typically have only partial knowledge, and/or are unable to articulate all their knowledge precisely and in the absence of sensory prompts, that we modelled our task as one where the expert offers information 'piecemeal', in reaction to the agent's latest actions (thereby making the expert's information relevant, or coherently related, to the agent's latest action, whether that action is asking a question, or performing an action in the domain). But while we support the expert being unable to impart all her knowledge in advance, we do not yet support that information being fallible. If the expert is fallible, then our agent would have to weigh the expert's advice against its own experience, and possibly revise its beliefs about the expert's abilities if they are consistently wrong.

There is also room to expand the range of expressibility with which the expert gives advice. For example, the expert could not only give the advice that one action is better than another, but also couple this with a partial causal explanation of *why* one action is better than another. As discussed throughout previous chapters, this may introduce further ambiguity due to the relative differences in each speaker's awareness, as it may not be clear whether such a causal explanation applies in all cases, or is conditional on some additional factor that the learning agent is unaware of.

Extending the Complexity of the Tasks: This thesis focussed primarily on discrete, finite-state decision problems in which variables became fully-observable the moment the agent became aware of them. However, in many real-world scenarios, a particular

factor can remain unobservable even after one becomes aware that it is relevant to the task. Extending such models of unawareness to incorporate the discovery of *hidden variables* is a significant challenge.

Another extension would be to consider tasks involving budgeted learning, where the agent knows it is only able to take a limited number of actions in each task, or where communication with the expert comes at a cost. Such work would likely involve evaluated the expected *value of information* associated with each piece of advice, which becomes difficult when the agent is potentially unaware of all possible outcomes it should evaluate over.

Strengthening the Reasoning Abilities of the Learning Agent In our current model, the agent only asks for advice when there is an *explicit conflict* between its current model and the partial descriptions it has gathered from domain and dialogue evidence so far. However, our agent could potentially be more pro-active, and ask about areas of the problem where it *suspects* it is unaware of something. The agent could achieve this by, for example, looking for structural signatures that indicate the location of a latent factor (Elidan *et al.*, 2000), or by asking about dependencies for which it has high uncertainty (Masegosa and Moral, 2013).

Appendix A

The language for partially describing Decision Networks

The model for learning the true DN (dn_+) in Chapter 4 made use of a language $\mathcal{L}$ for partially describing DNs. This language was used in two ways: to represent monotonic information from evidence, and as a language in which the learner and expert can communicate about (partial) information about the true DN. Its syntax and semantics are defined below.

In Chapter 5, we also use a similar language to for partially describing FMDPs. We have omitted a full specification of this related language to avoid repetition.

A.1 The syntax of the language $\mathcal{L}$

- Terms of the sort: $X, Y \ldots$ are *random variable* (RV) constants; Pa_Y, $\mathcal{X}$, $\mathcal{A}$, O, $scope(\mathcal{R})$, $\mathcal{B}$ are Sets of Random Variables (SRV) constants (denoting sets of random variables in the model). Where $\mathcal{Y}$ is an SRV constant, y is a partial-state assignment (PS) term, denoting a value assignment to each RV constant in Y; s is an atomic state (AS) term (denoting a full atomic state in the model). The language also includes RV variables and AS variables.
- If p is a PS term or an AS term, then it is a well-formed formula (WFF) in $\mathcal{L}$.
- If p is a PS or AS term and $\mathcal{X}$ is an SRV constant, then $p[\mathcal{X}]$ is a PS term in $\mathcal{L}$.

- If X is an RV term and $\mathcal{Y}$ is an SRV term, then $X \in \mathcal{Y}$ is a WFF in $\mathcal{L}$.
- If s is an AS term and n is a number, then $\mathcal{R}(s) = n$ and $\mathcal{R}(s) > n$ are both WFF in $\mathcal{L}$.
- Where ϕ, ψ are WFFs, so are $\neg\phi$, $\phi \wedge \psi$, $\exists s\phi$, $\forall s\phi$ (where s is an AS variable), $\exists V\phi$, $\forall V\phi$, and $?\lambda V\phi$ (where V is an RV variable).

Each model dn for interpreting $\mathcal{L}$ corresponds to a (unique) complete DN (see Section 4.1.1 for definition). Section A.2 then evaluates the formulae of $\mathcal{L}$ as partial descriptions of dn.

A.2 The semantics of $\mathcal{L}$

Let $dn = \langle \mathcal{X}^{dn}, \mathcal{A}^{dn}, Pa^{dn}, \theta^{dn}, \mathcal{R}^{dn} \rangle$ be a DN and g a variable assignment function.

- For an RV constant X, $[\![X]\!]^{\langle dn,g\rangle} = X$; similarly for SRV constants.[1] This ensures that $[\![\mathcal{A}]\!]^{\langle dn,g\rangle}$, $[\![\mathcal{B}]\!]^{\langle dn,g\rangle}$ and $[\![O]\!]^{\langle dn,g\rangle}$ denote sets of variables in the appropriate position in the probabilistic structure of dn.
- For an AS variable s, $[\![s]\!]^{\langle dn,g\rangle} = g(s)$ where $g(s) \in v(\mathcal{X}^{dn} \cup \mathcal{A}^{dn})$. For an RV variable V, $[\![V]\!]^{\langle dn,g\rangle} = g(V) \in \mathcal{X}^{dn} \cup \mathcal{A}^{dn}$.
- For an RV term a and an SRV term b, $[\![a \in b]\!]^{\langle dn,g\rangle} = 1$ iff $[\![a]\!]^{\langle dn,g\rangle} \in [\![b]\!]^{\langle dn,g\rangle}$.
- Where p is a PS term, $[\![p]\!]^{\langle dn,g\rangle} = p$.
- For an AS term s and number n, $[\![\mathcal{R}(s) = n]\!]^{\langle dn,g\rangle} = 1$ iff $\mathcal{R}_{dn}([\![s]\!]) = n$ (with $[\![\mathcal{R}(s) > n]\!]^{\langle dn,g\rangle}$ defined analogously).
- where p and q are AS or PS terms, $[\![p = q]\!]^{\langle dn,g\rangle} = 1$ iff $[\![p]\!]^{\langle dn,g\rangle} = [\![q]\!]^{\langle dn,g\rangle}$.
- Where p is an AS or PS term and $\mathcal{X}$ an SRV constant, $[\![p[\mathcal{X}]]\!]^{\langle dn,g\rangle} = [\![p]\!]^{\langle dn,g\rangle}\left[[\![\mathcal{X}]\!]^{\langle dn,g\rangle}\right]$ (i.e., the projection of the denotation of p onto the set of variables denoted by $\mathcal{X}$).
- For formulae ϕ, ψ: $[\![\phi \wedge \psi]\!]^{\langle dn,g\rangle} = 1$ iff $[\![\phi]\!]^{\langle dn,g\rangle} = 1$ and $[\![\phi]\!]^{\langle dn,g\rangle} = 1$; $[\![\neg\phi]\!]^{\langle dn,g\rangle} = 1$ iff $[\![\phi]\!]^{\langle dn,g\rangle} = 0$. Where s is an AS, $[\![\exists s\phi]\!]^{\langle dn,g\rangle} = 1$ iff there is a variable assignment function $g' = g[s/p]$ such that $[\![\phi]\!]^{\langle dn,g'\rangle} = 1$.

[1] If $X \notin \mathcal{C}_{dn} \cup \mathcal{A}_{dn}$, then $[\![X]\!]^{\langle dn,g\rangle}$ is undefined and Definition A.2 ensures that any formula ϕ featuring X is such that $dn \not\models \phi$; similarly for propositional terms p featuring a value of a variable that is not a part of dn.

- Where V is a RV variable and ϕ is a formula:
  $[\![\exists V\phi]\!]^{\langle dn,g\rangle} = 1$ iff there exists an RV constant X such that $[\![\phi[V/X]]\!]^{\langle dn,g\rangle} = 1$.
  $[\![?\lambda V\phi]\!]^{\langle dn,g\rangle} = \{\phi[V/X] : X \text{ is an RV constant and } [\![\phi[V/X]]\!]^{\langle dn,g\rangle} = 1\}$.

These interpretations yield a satisfaction relation in the usual way: $dn \models \phi$ iff there is a function g such that $[\![\phi]\!]^{\langle dn,g\rangle} = 1$.

Appendix B

Proof of Update Formulae (3.7)

Note, the derivation below corrects an error from the original Buntine (1991) paper.

The proof that:

$$\mathrm{BD}_t(X, Pa_X) = \mathrm{BD}_{t-1}(X, Pa_X)\frac{N^t_{i|j} + \alpha_{i|j} - 1}{N^t_{\cdot|j} + \alpha_{\cdot|j} - 1} \tag{3.7}$$

Follows from the definition of the multivariate-β function in terms of the Γ-function, and the recursive structure of Γ:

$$\beta(n_1, \ldots n_m) = \frac{\prod_{k=0}^m \Gamma(n_k)}{\Gamma(\sum_{k=0}^m n_k)} \tag{B.1}$$

$$\Gamma(x) = (x-1)\Gamma(x-1) \tag{B.2}$$

From equation (3.2), we have:

$$\mathrm{BD}_t(X, Pa_X) = P(Pa_X)\prod_{j \in v(Pa_X)} \frac{\beta(N^t_{0|j} + \alpha_{0|j}, \ldots, N^t_{m|j} + \alpha_{m|j})}{\beta(\alpha_{0|j}, \ldots, \alpha_{m|j})} \tag{3.2}$$

Lets assume that $d_t[X] = i$ and $d_t[Pa_X] = j$. The only difference between $BD_t(X, Pa_X)$ and $BD_{t-1}(X, Pa_X)$ is between counts $N^t_{i|j}$ and $N^{t-1}_{i|j}$ (specifically, that $N^{t-1}_{i|j} = N^t_{i|j} - 1$), so we can rewrite (3.2) as:

$$
\begin{aligned}
&\mathrm{BD}_t(X, Pa_X) \\
&= BD_{t-1}(X, Pa) \frac{\Gamma(\sum_{k=0}^{m} N_{k|j}^{t-1}) \prod_{k=0}^{m} \Gamma(N_{k|j}^{t})}{\Gamma(\sum_{k=0}^{m} N_{k|j}^{t}) \prod_{k=0}^{m} \Gamma(N_{k|j}^{t-1})} \\
&= \frac{\Gamma((\sum_{k=0}^{m} N_{k|j}^{t}) - 1)\Gamma(N_{i|j}^{t})}{\Gamma(\sum_{k=0}^{m} N_{k|j}^{t})\Gamma(N_{i|j}^{t} - 1)}
\end{aligned}
$$

Then, via (B.2), we have:

$$
\begin{aligned}
&\frac{\Gamma((\sum_{k=0}^{m} N_{k|j}^{t}) - 1)\Gamma(N_{i|j}^{t})}{\Gamma(\sum_{k=0}^{m} N_{k|j}^{t})\Gamma(N_{i|j}^{t} - 1)} \\
&= \frac{\Gamma((\sum_{k=0}^{m} N_{k|j}^{t}) - 1)(N_{i|j}^{t} - 1)\Gamma(N_{i|j}^{t} - 1)}{((\sum_{k=0}^{m} N_{k|j}^{t}) - 1)\Gamma((\sum_{k=0}^{m} N_{k|j}^{t}) - 1)\Gamma(N_{i|j}^{t} - 1)} \\
&= \frac{N_{i|j}^{t} - 1}{(\sum_{k=0}^{m} N_{k|j}^{t}) - 1}
\end{aligned}
$$

And therefore that:

$$
\mathrm{BD}_t(X, Pa_X) = \mathrm{BD}_{t-1}(X, Pa_X) \frac{N_{i|j}^{t} + \alpha_{i|j} - 1}{N_{\cdot|j}^{t} + \alpha_{\cdot|j} - 1} \tag{3.7}
$$

As required.

Appendix C

Task Specifications from Experiments

This chapter gives specifications of DNs used in Chapter 4's experiments. Tables C.1-C.4 correspond to the barley (18 variable), small (12 variable), medium (24 variable) and large (36 variable) DNs. Since all variables are binary, numbers the column $P(X = 1|Pa)$ just give probability that $X = 1$ for each possible assignment to $v(Pa)$ (Going from left to right, the first number in each row corresponds to the case where all parents have assignment 0, and the last number corresponds to the case where all parents have assignment 1). In addition, the numbers in the $\mathcal{R}(s)$ column give the reward received given the corresponding assignment to the variables in $scope(\mathcal{R})$.

The DNs were randomly generated by the following process: First, to generate a DN structure, the structural requirements were posed as a constraint programming problem (e.g that the variables must form a DAG, action variables cannot have parents, all variables must be connected to the reward node etc.). Then, using an off-the-shelf constraint programming library,[1] we solved for 100,000 solutions, then chose one randomly. Each set of parameters $\theta_{X|Pa_X}$ was then sampled from a dirichlet distribution with $\alpha_i = 1$ for all i.

For the sake of space, the specifications for the BNs in Chapter 3 and the FMDPs in Chapter 5 are not included here. However, full specifications of all BNs used in Chapter 3 can be found at `http://www.bnlearn.com/bnrepository/`, while full specifications for the *coffee* and *factory* FMDPs can be found at `https://cs.uwaterloo.ca/~jhoey/research/spudd/index.php`

[1] `https://developers.google.com/optimization/cp/`

Table C.1: barley DN (18 variables)

X	Pa_X	$P(X\|Pa_X)$
Soil Type	$\emptyset$	0.50
Temperature	$\emptyset$	0.50
Precipitation	$\emptyset$	0.50
Insect Prevalence	$\emptyset$	0.50
Local Concern	$\emptyset$	0.50
Nitrogen	Soil Type, Precipitation, Pesticide, Fertiliser	0.40, 0.60, 0.50, 0.70, 0.30, 0.50, 0.40, 0.60, 0.65, 0.85, 0.75, 0.95, 0.55, 0.75, 0.65, 0.85
Gross Crops	Harrow, Nitrogen, Grain	0.50, 0.40, 0.80, 0.70, 0.60, 0.50, 0.90, 0.80
Fungus	Temperature, Fungicide, Grain	0.20, 0.50, 0.02, 0.04, 0.30, 0.60, 0.03, 0.06
Weeds	Temperature, Harrow, Soil Type	0.20, 0.10, 0.02, 0.01, 0.30, 0.15, 0.03, 0.01
Infestation	Insect Prevalence, Pesticide	0.10, 0.50, 0.01, 0.05
Yield	Gross Crops, Fungus, Weeds, Infestation	0.20, 0.95, 0.10, 0.50, 0.10, 0.70, 0.05, 0.30, 0.05, 0.65, 0.01, 0.20, 0.01, 0.45, 0.01, 0.10
Protein	Nitrogen, Fertiliser, Grain	0.50, 0.90, 0.40, 0.80, 0.40, 0.80, 0.30, 0.70
Bad Press	Local Concern, Pesticide	0.00, 0.00, 0.01, 0.50

	$scope(\mathcal{R})$	$\mathcal{R}(s)$
	Yield, Protein, Fungus, Bad Press	0.75, 0.88, 0.88, 1.00, 0.50, 0.62, 0.62, 0.75, 0.25, 0.38, 0.38, 0.50, 0.00, 0.12, 0.12, 0.25

Table C.2: small DN: (12 Variables)

X	Pa_X	$P(X\|Pa_X)$
B1	$\emptyset$	0.50
O1	A4	0.23, 0.18
B2	$\emptyset$	0.84
B3	$\emptyset$	0.18
O3	A1, A3, B3	0.31, 0.89, 0.87, 0.15, 0.48, 0.21, 0.90, 0.06
B4	B2	0.45, 0.26
O4	A2, B1	0.78, 0.87, 0.12, 0.07
O2	B2, B4, O1	0.04, 0.75, 0.64, 0.07, 0.36, 0.62, 0.69, 0.17

	$scope(\mathcal{R})$	$\mathcal{R}(s)$
	O4, O2, O3	0.98, 0.14, 0.70, 0.35, 0.38, 0.98, 0.11, 0.97

Table C.3: medium DN (24 variables)

X	Pa_X	$P(X\|Pa_X)$
B8	$\emptyset$	0.05
B6	$\emptyset$	0.97
B7	$\emptyset$	0.01
B1	$\emptyset$	0.72
B4	$\emptyset$	0.08
B5	$\emptyset$	0.66
B2	$\emptyset$	0.78
B3	$\emptyset$	0.54
O3	A6, B4, B6	0.66, 0.14, 0.99, 0.83, 0.75, 0.71, 0.30, 0.12
O2	A5, A8, B5	0.26, 0.94, 0.67, 0.18, 0.04, 0.63, 0.87, 0.14
O6	A6, B3	0.87, 0.31, 0.48, 0.13
O4	A1, B8	0.27, 0.51, 0.12, 0.73
O8	A4, B2, O4	0.81, 0.51, 0.44, 0.44, 0.34, 0.82, 0.04, 0.56
O1	A2, A7, O3	0.03, 0.47, 0.74, 0.43, 0.59, 0.08, 0.60, 0.57
O7	B7, O6	0.83, 0.68, 0.27, 0.23
O5	A3, B1, O7	0.75, 0.30, 0.25, 0.35, 0.64, 0.30, 0.31, 0.31

	$scope(\mathcal{R})$	$\mathcal{R}(s)$
	O1, O8, O5, O2	0.47, 0.95, 1.00, 0.18, 0.26, 0.75, 0.38, 0.76, 0.60, 0.78, 0.02, 0.75, 0.40, 0.32, 0.30, 0.14

Table C.4: large DN (36 variables)

X	Pa_X	$P(X\|Pa_X)$
B10	$\emptyset$	0.43
O9	A12	0.53, 0.82
B11	$\emptyset$	0.59
O8	A2	0.87, 0.19
B12	$\emptyset$	0.42
B13	$\emptyset$	0.52
B14	$\emptyset$	0.76
B15	$\emptyset$	0.12
O2	A1, A6, A8	0.43, 0.89, 0.00, 0.18, 0.19, 0.76, 0.76, 0.67
B8	$\emptyset$	0.28
B9	$\emptyset$	0.26
B6	$\emptyset$	0.96
B7	$\emptyset$	0.27
B1	$\emptyset$	0.99
B4	$\emptyset$	0.99
B5	$\emptyset$	0.13
B2	$\emptyset$	0.77
B3	$\emptyset$	0.01
O3	B2, B9, O2	0.31, 0.21, 0.71, 0.65, 0.78, 0.17, 0.67, 0.92
O6	A3, B6, B11	0.28, 0.41, 0.55, 0.42, 0.93, 0.58, 0.44, 0.74
O5	A13, B3, B4	0.17, 0.66, 0.93, 0.62, 0.47, 0.08, 0.82, 0.86
O15	A2, A11, O8	0.92, 0.34, 0.25, 0.48, 0.43, 0.25, 0.52, 0.32
O14	A10, B5, B8	0.37, 0.91, 0.75, 0.86, 0.29, 0.33, 0.96, 0.84
O13	A4, A7, B1	0.37, 0.40, 0.74, 0.94, 0.67, 0.70, 0.66, 0.46
O7	B7, B15, O6	0.31, 0.69, 0.17, 0.56, 0.88, 0.40, 0.65, 0.91
O4	A14, O3, O9	0.31, 1.00, 0.59, 0.81, 0.08, 0.20, 0.02, 0.63
O12	B13, B14, O13	0.55, 0.71, 0.61, 0.22, 0.51, 0.90, 0.37, 0.52
O11	A5, B10, O5	0.73, 0.65, 0.42, 0.47, 0.42, 0.34, 0.97, 0.33
O10	A15, B12, O15	0.06, 0.80, 0.42, 0.36, 0.62, 0.97, 0.15, 0.29
O1	A9, O7, O11	0.48, 0.03, 0.14, 0.41, 0.16, 0.64, 0.22, 0.19

	$scope(\mathcal{R})$	$\mathcal{R}$
	O1, O4, O14, O10, O12	0.28, 0.19, 0.91, 0.71, 0.68, 0.57, 0.98, 0.86, 0.96, 0.54, 0.19, 0.20, 0.72, 0.61, 0.32, 0.65, 0.64, 0.12, 0.64, 0.67, 0.72, 0.27, 0.05, 0.07, 0.63, 0.09, 0.55, 0.45, 0.19, 0.02, 0.51, 0.13

Bibliography

Albrecht, S. V. and Ramamoorthy, S. (2016). Exploiting causality for selective belief filtering in dynamic Bayesian networks. *Journal of Artificial Intelligence Research*, **55**, 1135–1178.

Alchourrn, C. E., Grdenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *The journal of symbolic logic*, **50**(2), 510–530.

Alcobe, J. R. (2005). Incremental methods for Bayesian network structure learning. *Artificial Intelligence Communications*, **18**(1), 61–62.

Artzi, Y. and Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, **1**, 49–62.

Bartlett, M. and Cussens, J. (2017). Integer Linear Programming for the Bayesian network structure learning problem. *Artificial Intelligence*, **244**, 258–271.

Bengio, Y. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Besbes, O., Gur, Y., and Zeevi, A. (2014). Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in neural information processing systems*, pages 199–207.

Bielza, C., Gomez, M., and Shenoy, P. P. (2010). Modeling challenges with influence diagrams: Constructing probability and utility models. *Decision Support Systems*, **49**(4), 354–364.

Board, O. J., Chung, K.-S., and Schipper, B. C. (2011). Two models of unawareness:

Comparing the object-based and the subjective-state-space approaches. *Synthese*, **179**(1), 13–34.

Bos, J., Clark, S., Steedman, M., Curran, J. R., and Hockenmaier, J. (2004). Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1240. Association for Computational Linguistics.

Boutilier, C. and Goldszmidt, M. (1996). The frame problem and Bayesian network action representations. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 69–83. Springer.

Boutilier, C., Dearden, R., and Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial Intelligence*, **121**(1), 49–107.

Boutilier, C., Bacchus, F., and Brafman, R. I. (2001). UCP-networks: A directed graphical representation of conditional utilities. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 56–64. Morgan Kaufmann Publishers Inc.

Bramley, N. R., Dayan, P., Griffiths, T. L., and Lagnado, D. A. (2016). Formalizing Neuraths Ship: Approximate Algorithms for Online Causal Learning.

Buntine, W. L. (1991). Theory Refinement on Bayesian Networks. *CoRR*, **abs/1303.5709**.

Cadilhac, A., Asher, N., Lascarides, A., and Benamara, F. (2015). Preference change. *Journal of Logic, Language and Information*, **24**(3), 267–288.

Cakmak, M. and Thomaz, A. (2012). Designing robot learners that ask good questions. *Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction*.

Chen, C., Seff, A., Kornhauser, A., and Xiao, J. (2015). Deepdriving: Learning affordance for direct perception in autonomous driving. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2722–2730. IEEE.

Chouard, T. (2016). The Go Files: AI computer wraps up 4-1 victory against human champion. *Nature News*.

Coenen, A., Nelson, J. D., and Gureckis, T. M. (2017). Asking the right questions about

human inquiry. *OpenCoenen, Anna, Jonathan D Nelson, and Todd M Gureckis.Asking the Right Questions About Human Inquiry. PsyArXiv*, **13**.

Da Silva, B. C., Basso, E. W., Bazzan, A. L., and Engel, P. M. (2006). Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd international conference on Machine learning*, pages 217–224. ACM.

DARPA-BAA-16-53 (2016). Broad agency announcement on explainable artificial intelligence (xai).

Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational intelligence*, **5**(2), 142–150.

Degris, T. and Sigaud, O. (2010). Factored markov decision processes. *Markov Decision Processes in Artificial Intelligence*, pages 99–126.

Degris, T., Sigaud, O., and Wuillemin, P.-H. (2006). Learning the structure of factored markov decision processes in reinforcement learning problems. In *Proceedings of the 23rd international conference on Machine learning*, pages 257–264. ACM.

Dobnik, S., Cooper, R., and Larsson, S. (2012). Modelling language, action, and perception in type theory with records. In *International Workshop on Constraint Solving and Language Processing*, pages 70–91. Springer.

Doshi-Velez, F. (2009). The infinite partially observable Markov decision process. In *Advances in neural information processing systems*, pages 477–485.

Elidan, G., Lotner, N., Friedman, N., Koller, D., and others (2000). Discovering hidden variables: A structure-based approach. In *NIPS*, volume 13, pages 479–485.

Fagin, R. and Halpern, J. Y. (1987). Belief, awareness, and limited reasoning. *Artificial Intelligence*, **34**(1), 39–76.

Feinberg, Y. (2004). Subjective reasoning-games with unawareness.

Forbes, M., Rao, R. P. N., Zettlemoyer, L., and Cakmak, M. (2015). Robot Programming by Demonstration with situated spatial language understanding. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2014–2020.

Friedman, N. (1998). The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 129–138. Morgan Kaufmann Publishers Inc.

Friedman, N. and Goldszmidt, M. (1997). Sequential update of Bayesian network structure. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 165–174. Morgan Kaufmann Publishers Inc.

Friedman, N., Nachman, I., and Per, D. (1999). Learning bayesian network structure from massive datasets: the sparse candidate algorithm. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 206–215. Morgan Kaufmann Publishers Inc.

Grice, H. P. (1975). Logic and conversation. *1975*, pages 41–58.

Guestrin, C., Koller, D., Parr, R., and Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, **19**, 399–468.

Halpern, J. Y. and Rego, L. C. (2013). Reasoning about knowledge of unawareness revisited. *Mathematical Social Sciences*, **65**(2), 73–84.

Hansson, S. (1995). Changes in Preference. *Theory and Decision*.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning*, **20**(3), 197–243.

Heifetz, A., Meier, M., and Schipper, B. C. (2013). Dynamic unawareness and rationalizable behavior. *Games and Economic Behavior*, **81**, 50–68.

Herman, M., Gindele, T., Wagner, J., Schmitt, F., and Burgard, W. (2016). Inverse reinforcement learning with simultaneous estimation of rewards and dynamics. In *Artificial Intelligence and Statistics*, pages 102–110.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, **18**(7), 1527–1554.

Hobbs, J. R., Stickel, M. E., Appelt, D. E., and Martin, P. (1993). Interpretation as abduction. *Artificial intelligence*, **63**(1-2), 69–142.

Hoey, J., St-Aubin, R., Hu, A., and Boutilier, C. (1999). SPUDD: Stochastic Planning Using Decision Diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 279–288, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Howard, R. A. and Matheson, J. E. (2005). Influence diagrams. *Decision Analysis*, **2**(3), 127–143.

Hristov, Y., Penkov, S., Lascarides, A., and Ramamoorthy, S. (2017). Grounding Symbols in Multi-Modal Instructions. *arXiv preprint arXiv:1706.00355*.

Karni, E. and Viero, M.-L. (2013). "Reverse Bayesianism": A choice-based theory of growing awareness. *American Economic Review*, **103**(7), 2790–2810.

Karni, E. and Viero, M.-L. (2017). Awareness of unawareness: A theory of decision making in the face of ignorance. *Journal of Economic Theory*, **168**(C), 301–328.

Knox, W. B. and Stone, P. (2009). Interactively shaping agents via human reinforcement: The TAMER framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16. ACM.

Knox, W. B. and Stone, P. (2010). Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems.

Koller, D. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Kristensen, K. and Rasmussen, I. A. (2002). The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. *Computers and Electronics in Agriculture*, **33**(3), 197–217.

Laird, J. E., Gluck, K., Anderson, J., Forbus, K. D., Jenkins, O. C., Lebiere, C., Salvucci, D., Scheutz, M., Thomaz, A., and Trafton, G. (2017). Interactive task learning. *IEEE Intelligent Systems*, **32**(4), 6–21.

Lakkaraju, H., Kamar, E., Caruana, R., and Horvitz, E. (2017). Identifying Unknown Unknowns in the Open World: Representations and Policies for Guided Exploration. In *AAAI*, pages 2124–2132.

Leonetti, M., Iocchi, L., and Ramamoorthy, S. (2011). Reinforcement learning through global stochastic search in N-MDPs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 326–340. Springer.

Li, J. (2008). Modelilng Unawareness in Arbitrary State Spaces. SSRN Scholarly Paper ID 1151335, Social Science Research Network, Rochester, NY.

Liang, P. (2005). *Semi-supervised learning for natural language*. PhD Thesis, Massachusetts Institute of Technology.

Liebman, E., Zavesky, E., and Stone, P. (2017). Autonomous Model Management via Reinforcement Learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1601–1603. International Foundation for Autonomous Agents and Multiagent Systems.

Madigan, D., York, J., and Allard, D. (1995). Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique*, pages 215–232.

Masegosa, A. R. and Moral, S. (2013). An interactive approach for Bayesian network learning using domain/expert knowledge. *International Journal of Approximate Reasoning*, **54**(8), 1168–1181.

McCallum, A. K. and Ballard, D. (1996). *Reinforcement learning with selective perception and hidden state*. Ph.D. thesis, University of Rochester. Dept. of Computer Science.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., and others (2015). Human-level control through deep reinforcement learning. *Nature*, **518**(7540), 529–533.

Murphy, K. P. (2001). Active learning of causal Bayes net structure.

Nielsen, T. D. and Jensen, F. V. (2004). Learning a decision maker's utility function from (possibly) inconsistent behavior. *Artificial Intelligence*, **160**(1-2), 53–78.

Pearl, J. (2018). Theoretical Impediments to Machine Learning With Seven Sparks from the Causal Revolution. *arXiv preprint arXiv:1801.04016*.

Poole, D. (1993). Probabilistic Horn abduction and Bayesian networks. *Artificial intelligence*, **64**(1), 81–129.

Rafferty, A. N., LaMar, M. M., and Griffiths, T. L. (2015). Inferring learners' knowledge from their actions. *Cognitive Science*, **39**(3), 584–618.

Reddy, S., Tckstrm, O., Collins, M., Kwiatkowski, T., Das, D., Steedman, M., and

Lapata, M. (2016). Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, **4**, 127–140.

Rong, N. (2016). *Learning in the Presence of Unawareness*. Ph.D. thesis, Cornell University.

Russell, S. J. and Norvig, P. (2002). Artificial Intelligence: A Modern Approach (International Edition).

Savage, L. J. (1972). *The Foundations of Statistics*. Courier Corporation. Google-Books-ID: zSv6dBWneMEC.

Settles, B. (2009). Active learning literature survey. *Computer Sciences Technical Report*.

Silver, D. L. (2011). Machine lifelong learning: challenges and benefits for artificial general intelligence. In *International Conference on Artificial General Intelligence*, pages 370–375. Springer.

Siskind, J. M. (1996). A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, **61**(1-2), 39–91.

Suryadi, D. and Gmytrasiewicz, P. J. (1999). Learning models of other agents using influence diagrams. In *UM99 User Modeling*, pages 223–232. Springer.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, **10**(Jul), 1633–1685.

Teyssier, M. and Koller, D. (2005). Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 584–590. AUAI Press.

Thrun, S. and Pratt, L. (2012). *Learning to learn*. Springer Science & Business Media.

Tian, J. and He, R. (2009). Computing posterior probabilities of structural features in Bayesian networks. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 538–547. AUAI Press.

Torrey, L. and Taylor, M. (2013). Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference*

on Autonomous agents and multi-agent systems, pages 1053–1060. International Foundation for Autonomous Agents and Multiagent Systems.

Torrey, L., Shavlik, J., Walker, T., and Maclin, R. (2006). Relational skill transfer via advice taking. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*.

Utgoff, P. E., Berkman, N. C., and Clouse, J. A. (1997). Decision tree induction based on efficient tree restructuring. *Machine Learning*, **29**(1), 5–44.

van Otterlo, M. (2009). Intensional dynamic programming. A Rosetta stone for structured dynamic programming. *Journal of Algorithms*, **64**(4), 169–191.

Wood, F., Griffiths, T. L., and Ghahramani, Z. (2006). A non-parametric Bayesian method for inferring hidden causes. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 536–543. AUAI Press.

Wu, Z., Schrater, P., and Pitkow, X. (2018). Inverse POMDP: Inferring What You Think from What You Do. *arXiv preprint arXiv:1805.09864*.

Yasin, A. and Leray, P. (2013). Incremental Bayesian network structure learning in high dimensional domains. In *Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on*, pages 1–6. IEEE.

Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.