Learning Who to Trust: Policy Learning in Single-Stage Decision Problems with Unreliable Expert Advice

-

Research Proposal

Tamlin Love (1438243)
Supervised by Dr. Ritesh Ajoodha and Dr. Benjamin Rosman

Submitted to the Faculty of Science, University of the Witwatersrand
in partial fulfilment of the requirements for the degree of Master of Science

Abstract

The transfer of knowledge about a given decision problem from one learning system to another during training is a useful tool in the field of reinforcement learning, allowing for faster convergence to an optimal policy. However, existing methods in this regard typically make the strong assumption that the advising system (known as the expert) is infallible. The focus of this research is to remove this assumption and propose a learning system capable of receiving advice from multiple unreliable expert advisers; learning to assess the unreliability of each expert and to take this unreliability into account when making decisions.

Within the context of single-stage decision problems, we propose methods of modelling the unreliability of experts, using these models to act upon advice offered and updating these models given feedback from an environment. We argue that the proposed learning system can take advantage of multiple experts to improve the rate of convergence to an optimal policy, while still being robust against these experts being unreliable to some degree.

Contents

Chapter 1

Introduction

Suppose we have a doctor equipped to observe certain symptoms and prescribe certain treatments. A patient, who may or may not be exhibiting a number of these symptoms, visits said doctor asking whether or not they should be treated and what course of treatment they should receive. In addition to observing symptoms, the doctor is able to view the patient's medical history, which may include, among other things, whether or not the patient belongs to a population susceptible to a given disease and whether or not they have a very rare genetic condition. The doctor may prescribe none, one or both of two treatments. One of these treatments is effective in curing a given disease, but may also induce a harmful side-effect in patients with the aforementioned rare genetic condition, and thus the doctor will have to resort to the weaker, but less risky, second treatment in these cases.

These types of medical diagnosis problems, and other problems like them, have attracted attention from researchers looking to replace the doctor with a software agent for a long time, with the goal of making such medical diagnoses more comprehensive and more widely available [Lauritzen and Spiegelhalter 1988; Heckerman and Nathwani 1992; Kao *et al.* 2018]. These approaches typically rely on training on data which has been correctly labelled by domain experts, or else having a domain expert construct some model of the problem which is subsequently refined by learning from such data. The interaction between agent and domain expert typically stops after learning begins, despite the fact that an expert's continued assistance throughout the learning process may be useful. Indeed, learning with domain expert assistance is commonplace in the ways humans learn; for example by way of student-teacher or apprentice-master interactions.

There have been a number of attempts to integrate domain expert knowledge into the learning process [Murphy 2001; Knox and Stone 2009; Innes and Lascarides 2019]. Although it has been shown that integrating such knowledge can assist in the speed of learning [Torrey and Taylor 2013], these approaches have their own limitations; namely that they assume the domain expert to be infallible. This assumption is not typically true when dealing with human experts; for example, doctors frequently disagree with each other about diagnoses and treatments. For large problems, different software experts may also disagree about the best course of action, as they may have different approximations of the optimal policy due to factors such as the number of trials the expert is trained upon or the learning method used.

In this proposal, we aim to show that even when this strong assumption of expert infallibility is dropped, learning systems can still benefit from advice given. We propose a system in which one or more potentially unreliable domain experts can advise a software agent while it learns; where the agent maintains an estimate of the degree to which each expert is reliable, uses this estimate to make decisions, and uses the outcomes of its decisions to update these estimates of unreliability. Our contributions come

in the form of choosing how to model unreliability, how to act when given potentially unreliable advice, and how to update unreliability estimates given feedback from an environment.

To that end, the structure of this proposal is as follows. In this chapter we have introduced the problem area and provided motivation for our research. Chapter 2 describes the various concepts required to formulate our problem as well as methods relevant to our research. In Chapter 3, we use these concepts and methods to devise our proposed learning system and discuss the contributions we hope to make in this research. Finally, in Chapter 4, we propose a number of experiments to test the efficacy of our learning system and break our proposed methodology into deliverables, fitting them into a time plan spanning 13 months.

Chapter 2

Background and Related Work

In the previous chapter we presented the research area and introduced the problem this research attempts to solve. In this chapter, we discuss the concepts and methods in the literature that are relevant to this research. In Section 2.1, we introduce the reinforcement learning problem and define a single-stage decision problem. In Section 2.2, we discuss graphical models, namely Bayesian networks and influence diagrams, and discuss how they can model a single-stage decision-making environment. We also formally present our example medical diagnosis problem from Chapter 1 as an influence diagram. In Section 2.3, we discuss methods to learn and update an agent's model of the environment, represented as an influence diagram. Finally, in Section 2.4, we discuss ways in which expert advice can be integrated into learning how to solve such an environment.

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a field of machine learning in which decision-making entities, known as *agents*, learn how to interact with an environment in order to maximise some cumulative reward [Sutton and Barto 2018]. There are many types of RL problem. This proposal, however, concerns itself with *single-stage decision problems*, otherwise known as *contextual bandits* [Langford and Zhang 2007], with discrete states and actions. In this type of problem, the agent first observes some state $s \in S$, assigned by the environment, where S refers to the set of all states (otherwise known as the state-space). The agent selects some action $a \in A$, where A is the set of all actions (otherwise known as the action-space), and then receives some utility $r(s, a) \in \mathbb{R}$ from the environment. Each round of state observation, action selection and environment feedback is referred to as a *trial*. In this setting, a policy $\pi : S \to A$ is a function that maps a state to an action. The goal of the agent in this setting is, $\forall s \in S$, to learn the optimal policy

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ EU(\pi(s)|s), \tag{2.1}$$

where $EU(a|s)$ denotes the *expected utility* of choosing action a given state s. The expected utility of each state is typically not given, and must be learned by the agent. The method used in this research is discussed in detail in Section 2.2.2.

$$EU(s_t, a') - EU(s_t, a_t) \tag{2.2}$$

During learning, an agent is often required to balance acting according to its estimate of the optimal policy, referred to as *exploitation*, and acting sub-optimally to improve its estimate of sub-optimal actions, referred to as *exploration*. A common way of balancing exploitation and exploration is by means of an ϵ-greedy agent [Sutton and Barto 2018]. With probability $\epsilon > 0$, the agent acts randomly and with probability $1 - \epsilon$, the agent acts according to its estimate of the optimal policy.

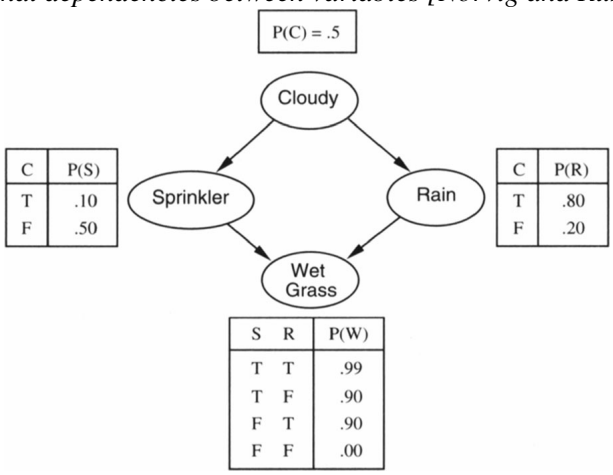
2.2 Graphical Models

As described in Equation 2.1, an RL agent learning a single-stage decision problem must calculate the expected utility of an action given a state in order to learn an optimal policy. In many approaches, every state and action are typically treated independently from each other. However, there is often some structure to the problem, and exploiting this structure may make it easier to learn an optimal policy. To achieve this, we factor the problem into a number of variables, and model these variables and the relationships between them using a graphical model.

2.2.1 Bayesian Networks

A Bayesian network (BN) is a compact representation of a set of variables $\mathcal{Z} = Z_1, ..., Z_N$ and the relationships between them that make up a joint distribution [Pearl 1988]. A BN consists of a structure $\mathcal{G}$ and parameters Θ. $\mathcal{G}$ is a directed acyclic graph (DAG) whose vertices represent the variables in $\mathcal{Z}$ and whose edges represent conditional dependencies between these variables. Thus if a directed edge exists from Z_i to Z_j, then Z_j is conditionally dependent on Z_i. More precisely, if $Pa^{\mathcal{G}}_{Z_i}$ denotes the parents of Z_i in $\mathcal{G}$ and $ND^{\mathcal{G}}_{Z_i}$ denotes the non-descendants of Z_i in $\mathcal{G}$, then Z_i is conditionally independent of $ND^{\mathcal{G}}_{Z_i}$ given $Pa^{\mathcal{G}}_{Z_i}$ [Koller and Friedman 2009]. The parameters $\theta_{Z_i} \in \Theta$ define the conditional probability distributions (CPDs) $\theta_{Z_i} = P(Z_i | Pa^{\mathcal{G}}_{Z_i})$. An example BN, complete with graphical structure and CPDs (given as tables), is provided in Figure 2.1.

Figure 2.1: *A famous example of a Bayesian network, illustrating the compact representation afforded by exploiting the conditional dependencies between variables [Norvig and Russell 1994].*



2.2.2 Influence Diagrams

Having discussed Bayesian networks, we now turn our attention to influence diagrams (IDs), otherwise known as decision networks, which extend Bayesian networks to include variables representing actions and utility, allowing them to be used in representing and solving some types of RL problems, including the single-stage decision problems discussed in Section 2.1.

Like Bayesian networks, influence diagrams consist of a DAG structure $\mathcal{G}$ and parameters Θ [Howard and Matheson 2005]. The variables $\mathcal{Z}$ that make up the nodes of $\mathcal{G}$ can be partitioned into three disjoint sets $\mathcal{Z} = \mathcal{X} \cup \mathcal{D} \cup \mathcal{U}$, where $\mathcal{X}$ denotes the set of *chance variables*, assignments of which function as states, $\mathcal{D}$ denotes the set of *decision variables*, assignments of which function as actions, and $\mathcal{U}$ denotes the set of *utility variables*, the sum of whose values equals the utility returned by the environment

[Koller and Friedman 2009]. For each decision variable $D \in \mathcal{D}$, the parent set $Pa_D^{\mathcal{G}}$ denotes the observations made before deciding on a value for D. In the context of single-stage decision problems, it is required that $Pa_D^{\mathcal{G}} \subseteq \mathcal{X}$ and that each $D \in \mathcal{D}$ has no decision variables as ancestors. In this case, we can partition the set of chance variables into two sets $\mathcal{X} = \mathcal{B} \cup \mathcal{O}$, where $\mathcal{B}$ denotes the set of chance variables observed before decision making, and $\mathcal{O}$ denotes the set of chance variables observed after decision making [Innes and Lascarides 2019].

In an influence diagram, every chance variable $X \in \mathcal{X}$ is associated with a CPD $\theta_X = P(X|Pa_X^{\mathcal{G}})$ and every utility variable $U \in \mathcal{U}$ is associated with a function $U(Pa_U^{\mathcal{G}})$ which maps to a real number [Koller and Friedman 2009]. The sum of each $U(Pa_U^{\mathcal{G}})$ is the total utility, which the agent seeks to maximise.

To illustrate these concepts, we formulate the medical diagnosis example from Chapter 1 as an influence diagram. Recall that the agent first observes the symptoms the patient is experiencing (chance variables *Symptom 1* and *Symptom 2*), as well as whether or not they belong to a susceptible population and whether or not they have a rare genetic condition (the variables *Susceptible* and *Rare Condition*). These are all observed before deciding whether or not to apply one or both of two available treatments (decision variables *Treatment 1* and *Treatment 2*) and are hence parents of the corresponding decision variables, forming the set $\mathcal{B}$. Whether or not the patient has the given disease is not observed directly and is rather inferred from the symptoms, and hence the variable *Disease* is not made a parent of any decision variables.

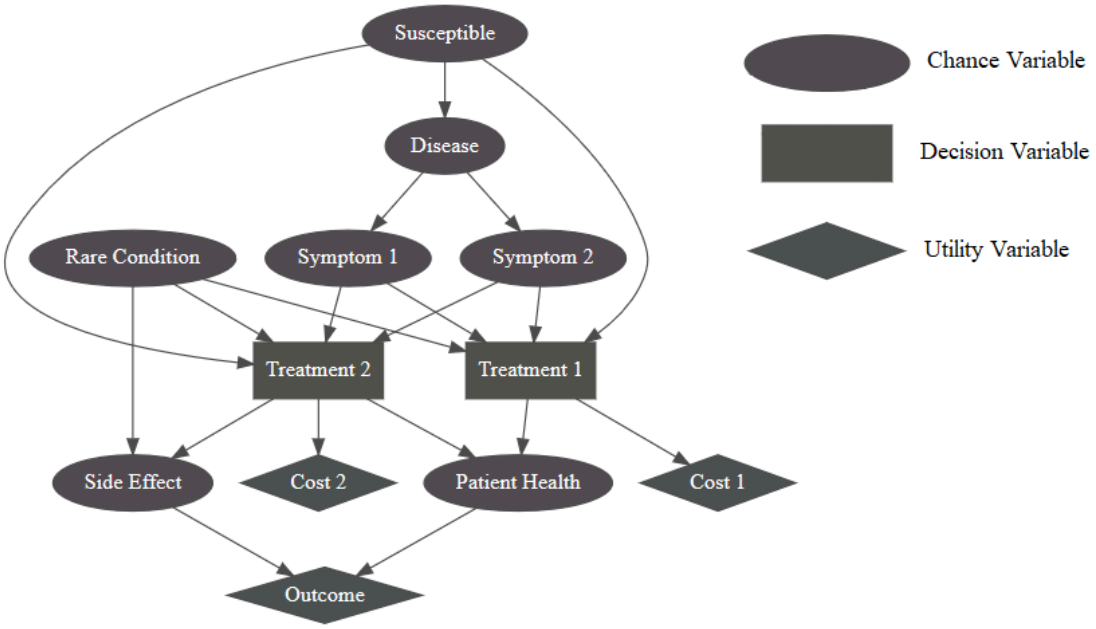
The decision to apply either or both treatments directly affects the patient's continued health, and thus the chance variable *Patient Health* is a child of both decision variables. The second treatment can, in patients with the rare genetic condition, exhibit harmful side effects, and thus the chance variable *Side Effect* is a child of both *Treatment 2* and *Rare Condition*. Together the variables *Patient Health* and *Side Effect* form the set $\mathcal{O}$. Finally, because we ultimately care about the patient's health and whether or not they have exhibited the harmful side effect, these two variables are made the parents of the utility variable *Outcome*, which rewards the agent for outcomes in which the patient is healthy and free of side effects, and penalises other outcomes. Additionally, we associate some costs (utility variables *Cost 1* and *Cost 2*) to the treatments to disincentivise prescribing treatments where they are not required. The final influence diagram is represented in Figure 2.2.

A policy π can be thought of as a set of CPDs for each decision variable $D \in \mathcal{D}$, and thus the problem of learning an optimal policy for a problem represented as an influence diagram can be reduced to learning these CPDs. Recall from Equation 2.1 that, in order to learn an optimal policy, the agent must be able to compute the expected utility of an action (or assignment of decision variables) given a state (or an assignment of chance variables). Let $\mathcal{I}_\pi$ be an ID representing a single-stage decision problem whose decision variables are assigned according to the policy π. Then the expected utility of $\mathcal{I}_\pi$ is given according to the following equation [Koller and Friedman 2009]

$$EU(\mathcal{I}_\pi) = \sum_{\mathcal{X} \cup \mathcal{D}} \left[\left(\prod_{X \in \mathcal{X}} P(X|Pa_X^{I_\pi}) \right) \left(\prod_{D \in \mathcal{D}} P_\pi(D|\mathcal{B}) \right) \left(\sum_{U \in \mathcal{U}} U(Pa_U^{I_\pi}) \right) \right]. \tag{2.3}$$

The first term in this equation is the product of each chance variable CPD factor, the second term is the product of each decision variable factor associated with the policy π, and the third term is the total utility that results from adding the value of each utility variable [Koller and Friedman 2009]. Using Equation 2.3, we can calculate the optimal policy for a given structure and set of parameters by means of the variable elimination algorithm, which operates by determining optimal decisions for each decision variable working backwards from the utility variables [Zhang and Poole 1994]. Thus if we can learn the best influence diagram to fit the environment, we can learn the optimal policy for that environment.

Figure 2.2: *An example medical diagnosis influence diagram.*



2.3 Learning Graphical Models

Given a set of trials $D_t = [d_0, ..., d_t]$, where every d_i denotes an assignment of $\mathcal{Z}$, we wish to learn the best-fitting influence diagram in order to learn an optimal policy for the environment. This problem is two-fold; we must learn a graph structure $\mathcal{G}_t$ that maximises $P(\mathcal{G}_t|D_t)$ and a set of parameters Θ_t that maximises $P(\Theta_t|\mathcal{G}_t, D_t)$. Additionally, as our data grows incrementally with every new trial, it would be prudent to employ methods that incrementally update our estimates of the model structure and parameters.

We therefore begin by addressing the problem of structure learning in graphical models. A common approach to the problem is a family of methods referred to as score-based structure learning methods. These methods assign a score to each candidate structure and then, by means of various optimisation techniques, find a structure that maximises this score [Koller and Friedman 2009]. A common scoring metric in these methods is the *Bayesian Dirichlet* score (BD score), defined as

$$BD(\mathcal{G}, D_t; n'_{1...N}) = \prod_{i=1}^{N} BD_{X_i}(\mathcal{G}, D_t; n'_{1...N}) \tag{2.4}$$

where

$$BD_{X_i}(\mathcal{G}, D_t; n'_{1...N}) = P(\mathcal{G}) \prod_{j=1}^{q_i} \left(\frac{\Gamma(n'_{ij})}{\Gamma(n'_{ij} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(n'_{ijk} + n_{ijk})}{\Gamma(n'_{ijk})} \right), \tag{2.5}$$

and where N is the number of variables in $\mathcal{G}$, q_i is the number of possible combinations of $Pa^{\mathcal{G}}_{X_i}$, r_i is the size of the domain of X_i, and $\Gamma(x)$ denotes the gamma function [Heckerman *et al.* 1995]. The n_i parameters are counts of assignments in D_t, with n_{ijk} denoting the number of occurrences where X_i takes its kth value out of r_i and $Pa^{\mathcal{G}}_{X_i}$ takes its jth value out of q_i, and $n_{ij} = \sum_{k=1}^{r_i} n_{ijk}$. Similarly, the n'_i parameters denote pseudo-counts used to weight the prior against the data, with $n'_{ij} = \sum_{k=1}^{r_i} n'_{ijk}$. This score balances fit to data with model complexity; preferring simpler structures with little data but allowing more complicated structures as the size of the data set grows [Koller and Friedman 2009].

In practice, calculating the BD score can be computationally intractable due to the large number of hyperparameters. A common special case of the BD score, known as the *Bayesian Dirichlet equivalent uniform* score (BDeu score) is defined by setting $n'_{ijk} = \frac{n'}{r_i q_i}$ [Heckerman *et al.* 1995], as follows

$$BDeu(\mathcal{G}, D_t; n') = \prod_{i=1}^{N} BDeu_{X_i}(\mathcal{G}, D_t; n') \tag{2.6}$$

where

$$BDeu_{X_i}(\mathcal{G}, D_t; n') = P(\mathcal{G}) \prod_{j=1}^{q_i} \left(\frac{\Gamma(\frac{n'}{q_i})}{\Gamma(\frac{n'}{q_i} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\frac{n'}{r_i q_i} + n_{ijk})}{\Gamma(\frac{n'}{r_i q_i})} \right). \tag{2.7}$$

Using this score reduces the number of hyperparameters to just one: n', the equivalent sample size. In practice, the logarithm of the score is usually taken to improve ease of calculation.

The BDeu score can be reformulated as an update rule that is incrementally updated after every new trial

$$BDeu_{X_i,t}(\mathcal{G}_t, D_t; n') = BDeu_{X_i,t-1}(\mathcal{G}_{t-1}, D_{t-1}; n') \frac{n^t_{ijk} + \frac{n'}{r_i q_i} - 1}{\sum_{l=1}^{r_i} \left(n^t_{ijl} \right) + \frac{n'}{q_i} - 1}, \tag{2.8}$$

where, during trial t, X_i takes the kth value out of r_i and $Pa^{\mathcal{G}_t}_{X_i}$ takes the jth value out of q_i [Innes and Lascarides 2019].

There are a variety of methods to find the maximum scoring structure, but the method most relevant to this research is that outlined by Innes and Lascarides [2019]. The first part of this method involves constructing *reasonable parent lattices* $\mathcal{P}_{X_i}$, data structures containing all parent sets under consideration for a given variable, for each variable X_i, as in Buntine [1991], by gradually combining high-scoring sets, starting at $\varnothing$, and eliminating low-scoring sets. Then, as in Bartlett and Cussens [2017], the method proceeds to find $\mathcal{G}$ via the linear program

$$\begin{aligned} \max & \sum_{i=1}^{N} \sum_{Pa^{\mathcal{G}}_{X_i} \in \mathcal{P}_{X_i}} I(Pa^{\mathcal{G}}_{X_i} \rightarrow X_i) ln(BD(X_i | Pa^{\mathcal{G}}_{X_i}; n'_i)) \\ \text{such that } & \sum_{Pa^{\mathcal{G}}_X \in \mathcal{P}_X} I(Pa^{\mathcal{G}}_X \rightarrow X) = 1 \forall X \\ & \sum_{X \in C, Pa_X \cap C = \varnothing} I(Pa^{\mathcal{G}}_X \rightarrow X) \geq 1 \forall C \subseteq \mathcal{Z} \\ & I(Pa^{\mathcal{G}}_X \rightarrow X) \in \{0, 1\} \forall X, Pa^{\mathcal{G}}_X, \end{aligned} \tag{2.9}$$

where $I(Pa^{\mathcal{G}}_X \rightarrow X)$ denotes whether or not $Pa^{\mathcal{G}}_X$ was chosen as the parent set for X. The constraints on the linear program ensure that every X has only one parent set and that the resulting structure is a DAG [Innes and Lascarides 2019].

After learning the structure $\mathcal{G}$, the problem of learning the parameters Θ for the chance variables is relatively straightforward. We can achieve this using the following expectation value

$$\mathbb{E}(\theta_{ijk} | \mathcal{G}, D_t) = \frac{n_{ijk} + \frac{n'}{r_i q_i}}{\frac{n'}{q_i} + \sum_{l=1}^{r_i} n_{ijl}}, \tag{2.10}$$

where i, j and k retain their meanings from Equations 2.4 to 2.8 above [Murphy 2001].

Learning the utility functions for each utility variable is also simple, as it merely requires setting

$$U(\zeta_t) = \frac{\sum_{d_\tau \in D_t, val_\tau(Pa_U^{\mathcal{G}})=\zeta_t} U_\tau}{n_{val_\tau(Pa_U^{\mathcal{G}})=\zeta_t}}, \tag{2.11}$$

where U_τ is the value of U observed in trial τ, $val_\tau(Pa_U^{\mathcal{G}})$ is the assignment of U's parents in trial τ, ζ_t is their assignment in trial t and $n_{val_\tau(Pa_U^{\mathcal{G}})=\zeta_t}$ is the number of times $val_\tau(Pa_U^{\mathcal{G}}) = \zeta_t$ has been true in D_t. In other words, $U(\zeta_t)$ is set to the expected value of the utility function whenever U's parents take the values in ζ_t. For unseen assignments, the utility can default to 0 [Innes and Lascarides 2019].

2.4 Incorporating Expert Advice

Having discussed single-stage decision problems and the ways in which they can be modelled and solved, we now turn our attention to the problem of incorporating expert advice into the learning process. As discussed in Chapter 1, knowledge imparted by an expert advisor, either a human or another agent trained on the same environment, can be used to improve convergence upon an optimal policy. The challenge here is how to incorporate expert advice into the framework of reinforcement learning and the learning of graphical models.

Some early attempts in learning models, such as Lauritzen and Spiegelhalter [1988] and Buntine [1991], involve eliciting knowledge from human experts and using this to build some prior model which subsequent data is used to refine. In these methods, the experts are only involved in the early stages of the learning process and cannot intervene once the agent has begun to refine the model with the data. In other methods, such as in Murphy [2001] and Masegosa and Moral [2013], an agent learns a model structure with the assistance of an expert, whom the agent can query about specific components of the model.

Others have the experts intervene during the learning of the policy. For example, in the work by Knox and Stone [2009], human experts can provide reward feedback to the agent as it learns; a process known as *shaping*. In work by Torrey and Taylor [2013], an expert with a sufficiently good policy π can advise an agent on which action to take exactly n times, and must budget this advice accordingly. Innes and Lascarides [2019] use an expert advisor to reveal state variables and actions that the agent was previously unaware of (termed *unawareness*), with the expert providing advice as to which action to perform in a given state and the agent using seeming contradictions in said advice to elicit information about unseen state variables.

Interactions between experts and agents can take many forms, including the reward shaping of Knox and Stone [2009] and learning from demonstration (*inverse reinforcement learning*) [Ng *et al.* 2000]. In this research, however, we focus on *action advice*, where, for a given state, the expert suggests an action to take. This type of advice requires only that the expert and agent share a common action set, thus requiring minimal similarity between the two entities [Torrey and Taylor 2013]. Given a state s_t, action advice takes the form

$$EU(a|s_t) \geq EU(a'|s_t)\ \forall a' \in A. \tag{2.12}$$

When this advice is given differs from work to work. Torrey and Taylor [2013] have the expert give the advice before the agent acts, whereas Innes and Lascarides [2019] have the expert advise the agent after acting. In both approaches, however, emphasis is placed on limiting agent-expert interaction, and thus conditions are placed upon when interactions can take place.

The simplest approaches to limiting interactions include limiting the number of interactions [Torrey and Taylor 2013] or using the constraint

$$t - t' > \mu, \tag{2.13}$$

where t is the current trial and t' is the last trial at which advice was offered, ensuring at least μ trials have occurred between these two trials [Innes and Lascarides 2019].

Torrey and Taylor [2013] also present the idea of *state importance*, which measures how important a state is based on the range of utility an agent can receive in that state. The importance of a state is given as

$$Im(s) = \max_a EU(a|s) - \min_a EU(a|s). \tag{2.14}$$

This captures the idea that a state is important if there is large room for error. When giving advice before the agent acts, and the expert knows at time t that the agent will perform action a (either because the agent has announced this or because the expert has predicted this), the expert will advise action a' if the following conditions are true

$$Im(s_t) \geq \gamma \text{ and } a' \neq a, \tag{2.15}$$

where γ is some threshold. Innes and Lascarides [2019] have a similar condition, based on the difference in expected utility between the expert's optimal actions a'_i and the actions a_i taken by the agent

$$\sum_{t' \leq i \leq t} \frac{EU(a'_i|s_i) - EU(a_i|s_i)}{t - t'} > \gamma, \tag{2.16}$$

with the additional condition

$$EU(a_t|s_t) \geq r_t, \tag{2.17}$$

which ensures the expected utility of the suggested action is not less than the reward the agent actually received.

In the aforementioned approaches, with the exception of Masegosa and Moral [2013], the expert is assumed to be infallible, and thus the agent can simply do as advised. Masegosa and Moral [2013] do allow for an explicitly unreliable expert, but the agent's model of the expert's reliability does not change as evidence of the expert's reliability builds up. Thus, in the next chapter, we address methods by which a notion of unreliability can be incorporated into learning systems with expert advice, and how this can allow for many expert advisers to be involved in the learning process at the same time.

Chapter 3

Methodology

In the previous chapter we defined a number of concepts at the core of this research. In this chapter, we formally state our research question and describe our proposed learning system. The outline of this chapter is as follows. In Section 3.1, we state the aims of this proposal and formally state the problem we are attempting to solve. In Section 3.2, we present a number of hypotheses to be tested in this research, and state a number of assumptions we make when testing these hypotheses. In Section 3.3, we describe the learning process and present our contributions.

3.1 Introduction and Problem Statement

We begin by discussing the problem this research addresses. We have some single-stage decision problem, such as the medical example described in Section 2.2.2. Within this environment, we have an RL agent attempting to learn some optimal policy. Assisting in this regard are one or more expert advisers that help the agent by offering it action advice in some trials (see Section 2.4). Recall that this advice takes the form

$$EU(a|s_t) \geq EU(a'|s_t)\ \forall a' \in A. \tag{3.1}$$

If the experts are assumed to be reliable in their advice, then the agent simply needs to follow the advice whenever applicable. For example, after being given the advice in Equation 3.1, whenever the agent observes the state s_t, it knows to choose the action a. However, if this assumption is lifted and the experts are assumed to be unreliable, simply following the experts' advice unquestioningly is no longer guaranteed to converge to an optimal policy.

At this point we formally define what we mean by reliable and unreliable advice. Suppose at time t the agent receives action advice of the form in Equation 3.1. In other words, the expert advises the agent that, given the observation s_t, the action a is expected to return the highest utility. For the advice to be considered *reliable*, Equation 3.1 must be true. Otherwise, the advice is said to be *unreliable*. If the advice is true $\forall a \in A$, $\forall s_t \in S$ and $\forall t$ (i.e. the expert always gives reliable advice), the expert uttering the advice is said to be *reliable*. Otherwise, the expert is said to be *unreliable*.

It has been shown that agents learning a policy for a single-stage decision problem with an expert advisor can converge to an optimal policy faster than agents learning a policy by themselves, provided that the expert is reliable [Torrey and Taylor 2013]. However, for an expert to be reliable, it must have access to some *ground truth* policy; an assumption that cannot hold for all problems.

In problems where a *ground truth* policy is not known, however, we need not do away with an expert advisor entirely. Indeed, an expert advisor whose advice is correct 90% of the time is likely to still be of

value to an agent. Conversely, however, an expert advisor whose advice is correct only 10% of the time is likely to be more a hindrance than a help to an agent. In both cases the experts are unreliable, but they are unreliable to different degrees.

Thus the problem we face is how to incorporate expert advice from one or more unreliable experts in such a fashion as to improve convergence to an optimal policy when the expert's advice is more often than not correct, while still not being hindered by experts whose advice is more often than not incorrect. The aim of this research, therefore, is to create a learning system that can take advantage of multiple experts to improve the rate of convergence to an optimal policy in a single-stage decision problem, while still being robust against these experts being unreliable to some degree.

3.2 Research Question

3.2.1 Assumptions

Having presented the problem in Section 3.1, we now further define our problem by introducing a number of assumptions.

1. As mentioned in Section 2.1, the type of RL problem addressed in this research is restricted to a single-stage decision problem.
2. We assume the state-space S can be factored into a number of state variables $X \in \mathcal{X}$. This allows us to make use of the graphical models presented in Section 2.2.
3. The agent is fully aware of the state-space S and action-space A at the start of learning. In other words, our learning system does not incorporate unawareness (as described by Innes and Lascarides [2019]). We do assume, however, that the agent will have to learn the CPDs of each state variable.

3.2.2 Research Hypotheses

Given the problem and research aims outlined in Section 3.1, we propose the following hypotheses to be tested given the learning system we shall outline in Section 3.3.

1. An agent learning with the advice of one or more reliable experts should converge to an optimal policy faster than an agent learning by itself.
2. An agent learning with the advice of one or more unreliable experts should converge to an optimal policy faster than an agent that incorporates the same advice and assumes it to be reliable.
3. When learning an internal model and making decisions, an agent learning with the advice of one or more unreliable experts should learn to rely more on experts that are sufficiently reliable and should learn to disregard advice from experts that are more often than not unreliable.

The first hypothesis is well supported in the literature. As there exist learning methods that incorporate reliable expert advice and that can converge to an optimal policy faster than agents that do not incorporate such advice, notably those by Innes and Lascarides [2019] and Torrey and Taylor [2013], it is sufficient for our learning system to not perform worse than these existing implementations.

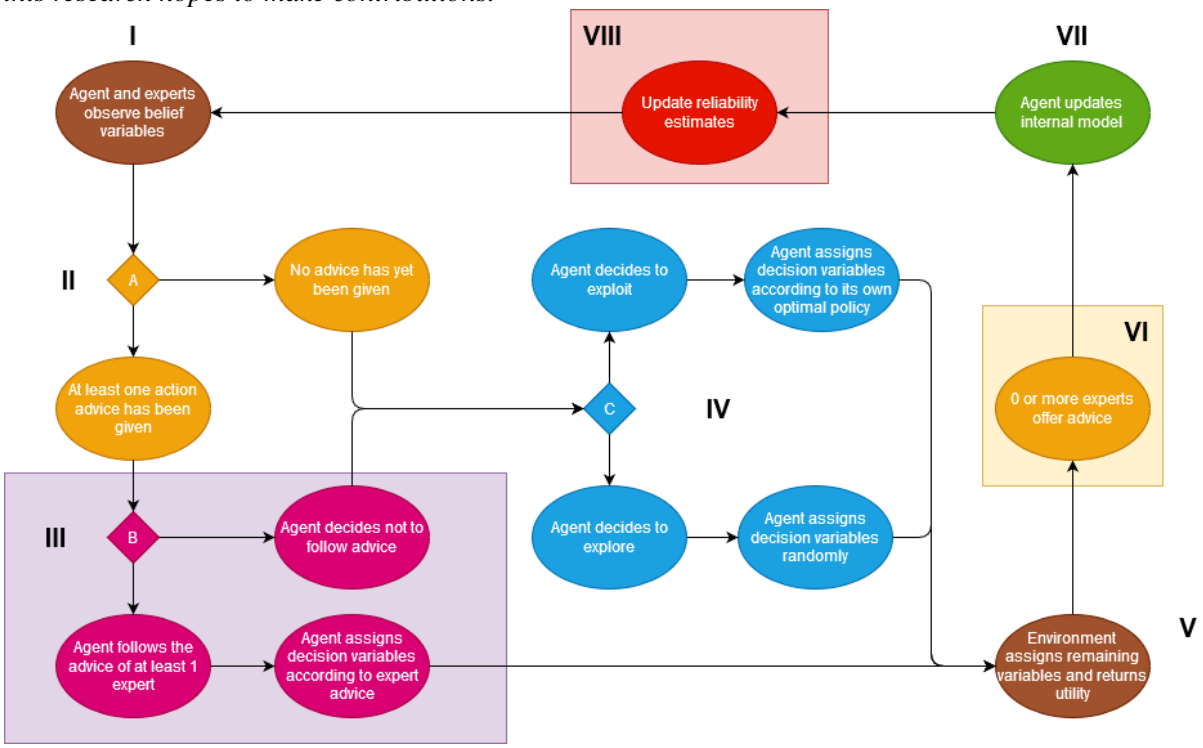
The second hypothesis merely states that, in the presence of unreliable advice, our learning system should converge to an optimal policy faster than an agent that incorporates the advice under the naïve assumption that the advice is reliable. If this hypothesis holds true, our agent can be considered robust against unreliable advice.

The third hypothesis is where the heart of this proposal lies. Our learning system should be able to learn to rely more on experts that, more often than not, offer correct advice, while learning to ignore the advice of experts who are more often than not incorrect. In order to do this, the agent must model the reliability of each expert and must update these estimates of reliability after each trial. These components of our learning system are discussed in detail in Sections 3.3.1 and 3.3.3 respectively.

3.3 Methodology

Having discussed the problem we are trying to solve and stated the aims of our research, we now propose a learning system to meet these aims. This learning system consists of a few actors: an agent and an environment (see Section 2.1 for descriptions of both), as well as one or more experts (see Section 2.4). The process by which these actors interact and by which the agent learns an optimal policy is depicted in Figure 3.1, and is described below.

Figure 3.1: *The proposed learning system process. Components enclosed in boxes denote areas in which this research hopes to make contributions.*



Before learning starts, the agent maintains some initial reliability estimate for each expert (see Section 3.3.1) and some initial model of the environment, modelled by an influence diagram (see Section 2.2.2). After initialisation, the first trial begins and the agent learns according to the following process.

In step **I**, the agent and the experts observe the chance variables $\mathcal{B} \subseteq \mathcal{X}$ set by the environment that form the state s_t.

In step **II**, the process branches depending on whether or not any advice has been given for this state in any previous trials. If no advice has yet been given, we proceed to step **IV**. Otherwise, we proceed to step **III**.

In step **III**, having received some advice as to what action to take in state s_t, the agent must decide whether or not to follow the advice it has been given and which advice, if any contradictions are present, to take. This decision is discussed in detail in Section 3.3.2. If the agent does not follow any advice, we proceed to step **IV**. Otherwise, having decided on an action to take, we proceed to step **V**.

In step **IV**, for whatever reason the agent is not following any advice, and thus behaves as an ϵ-greedy agent, as described in Section 2.1. After choosing an action, we proceed to step **V**.

In step **V**, after the agent acts, the environment assigns values to the remaining chance variables $\mathcal{O} \subseteq \mathcal{X}$ and to the utility variables $\mathcal{U}$. After this, we proceed to step **VI**.

In step **VI**, the experts must decide whether or not to advise the agent. We propose using conditions 2.13, 2.16 and 2.17 discussed in Section 2.4, with the additional constraint

$$I(a'_i \neq \varnothing)I(s_t \neq s_i) = 1 \; \forall i \in [0, ..., t-1], \tag{3.2}$$

where $I(a'_i \neq \varnothing) = 1$ if advice was given in trial i and equals 0 otherwise, and $I(s_t \neq s_i) = 1$ if $s_t \neq s_i$ and equals 0 otherwise. This constraint ensures that the expert has not advised the agent in this state before, as the advice would not change and thus giving the same advice in the same state twice would not add any information. After each expert does or does not advise the expert, we proceed to step **VII**.

In step **VII**, the agent updates the influence diagram it maintains as a model of the environment, using the update rules discussed in Section 2.3. After this, we proceed to step **VIII**.

In step **VIII**, the agent updates its estimates for the reliability of each expert, based on advice it has received thus far. This update is discussed in detail in Section 3.3.3. After this, either a new trial begins, in which case we proceed to step **I**, or the learning ends.

Having discussed the overall learning process, we now describe in detail each of the proposed contributions of this research (steps **III**, **VI** and **VIII** in Figure 3.1).

3.3.1 Modelling Unreliability

As mentioned in Section 3.2.2, in order for the agent to learn which experts to listen to and which to disregard, the agent must maintain some model of each of the experts' reliability. We now turn our attention to the problem of modelling expert reliability.

Recall from Section 3.1 that experts can be unreliable to different degrees. It seems reasonable then to attempt to model this degree of unreliability in each expert. This degree of unreliability ranges from an expert that is totally unreliable (that is, their advice is never correct) to an expert that is totally reliable (that is, their advice is always correct).

Given the above reasoning, we propose a method of modelling expert unreliability. We assume that an expert's unreliability is uniform across all decisions and states in the problem. Under this assumption, we model the unreliability $\rho \in [0, 1]$ as the expected value of a beta distribution $Beta_x[\alpha, \beta]$, where

$$Beta_x[\alpha, \beta] = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \tag{3.3}$$

the shape of which is controlled by parameters α and β [Owen 2008]. We choose a beta distribution to enforce that the estimation of unreliability be constrained between 0 (denoting a totally unreliable expert) and 1 (denoting a totally reliable expert).

Thus the expected value of the distribution is

$$\rho = \mathbb{E}[x] = \frac{\alpha}{\alpha + \beta}. \tag{3.4}$$

Extending this model to multiple experts is straightforward; for each expert $e^{(i)}$ the agent maintains a distribution $Beta_x[\alpha^{(i)}, \beta^{(i)}]$ from which the unreliability $\rho^{(i)}$ is calculated according to Equation 3.4.

3.3.2 Making Decisions

At trial t, a subset E^t of experts has offered advice to the agent (step **VI** in Figure 3.1) in trials $0, ..., t$ given the state s_t. Given this advice, we now address how the agent makes decisions. We note that there are three cases conditioned on the cardinality of E^t, and will thus address each case in turn.

In the first case, $E^t = \varnothing$. In other words, no experts offered advice for s_t. In this case, the agent cannot act upon advice it does not receive and hence acts as an ϵ-greedy agent (step **IV** in Figure 3.1).

In the second case, $|E^t| = 1$. In other words, a single expert has advised the agent on s_t. In this case, we propose using the agent's estimate of the experts unreliability as a parameter to decide whether or not to follow the advice, functioning similarly to the ϵ parameter in an ϵ-greedy agent. With probability ρ the agent acts according to the expert's advice and with probability $1 - \rho$ the agent acts as an ϵ-greedy agent.

In the third case, $|E^t| > 1$. In other words, more than one expert has advised the agent on s_t. We propose two methods of decision making in this case. In the first method, the agent chooses the expert in E^t with the highest ρ and then acts as if they are the only expert advising them, thus reducing the problem to the second case detailed above.

This method, although simple, eliminates the information provided by the other, less reliable experts. Consider, for example, a situation in which the most reliable expert advises taking action a, but several other, slightly less reliable experts advise taking action a'. In this situation, the consensus between multiple experts may be worth more than the advice of a single expert, even if they are generally more reliable, a phenomenon referred to as the "wisdom of the crowd" [Yi *et al.* 2012]. In another example, suppose that a very unreliable expert advises taking action a. Here the advice acts as adversarial advice; we know that the expert is most likely wrong, and hence the agent should be less likely to choose action a over other actions.

With these examples in mind, the second method we propose is designed to probabilistically aggregate all advice offered, in the vein of work by Burke and Klein [2020]. Let C denote the optimal action given state s_t and v_i denote the advice offered by expert $e^{(i)} \in E^t$, with $v_{1...I}$ denoting all advice offered for s_t. We therefore wish to find the probability $P(C = a_j | v_{1...I})$ for each $a_j \in A$. To do this, we employ Bayes's rule as follows

$$P(C = a_j | v_{1...I}) = \frac{P(v_{1...I} | C = a_j) P(C = a_j)}{\sum_{k=0}^{|A|} P(v_{1...I} | C = a_k) P(C = a_k)}, \tag{3.5}$$

which, assuming each expert's advice is given independently of the other experts, reduces to

$$P(C = a_j | v_{1...I}) = \frac{\prod_{i=0}^{|E^t|} P(v_i | C = a_j) P(C = a_j)}{\sum_{k=0}^{|A|} \prod_{i=0}^{|E^t|} P(v_i | C = a_k) P(C = a_k)}. \tag{3.6}$$

Finally, under the assumption of a uniform prior probability of each action being correct, we have the following expression

$$P(C = a_j | v_{1...I}) = \frac{\prod_{i=0}^{|E^t|} P(v_i | C = a_j)}{\sum_{k=0}^{|A|} \prod_{i=0}^{|E^t|} P(v_i | C = a_k)}. \tag{3.7}$$

We now turn our attention to the calculation of the likelihood term $P(v_i | C = a_k)$. Recall that the expert's advice is reliable if it is correct and unreliable if it is incorrect. Further recall that $\rho^{(i)}$ models

the probability of an expert's advice being correct. We now make the assumption that if the expert is incorrect in this trial, they have equal probability of selecting any other action apart from the optimal one, as in Masegosa and Moral [2013]. From this, it follows that

$$P(v_i|C = a_k) = \begin{cases} \rho^{(i)} & v_i = a_k \\ 1 - \rho^{(i)} & v_i \neq a_k. \end{cases} \tag{3.8}$$

Substituting Equation 3.8 into Equation 3.7, we can calculate the posterior probability of each action in A being optimal. Let a^* be the action that maximises $P(C = a_j|v_{1...I})$. Then we propose that, with probability $P(C = a^*|v_{1...I})$ the agent should take action a^*. Otherwise, the agent should behave as an ϵ-greedy agent.

3.3.3 Updating Unreliability Estimates

During the course of trial t, the agent observed the state s_t and received advice from a subset E^t of all the experts advising it in the trials $0, ..., t$. For each $e^{(i)} \in E^t$ the action $a^{(i)} \in A$ was advised. The agent then chose to take the action a_t, according to the methods discussed in Section 3.3.2, and subsequently received a total reward r_t, which it used to update its internal model. The agent must now update the reliability estimates of each of the experts in E^t. We propose the following method to achieve this. We begin by defining the following optimal action function

$$J(a, s_t) = \begin{cases} 1 & EU(a|s_t) \geq EU(a'|s_t) \forall a' \in A \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

In other words, $J(a, s_t) = 1$ if and only if the action a maximises $EU(a|s_t)$. Intuitively, for each expert $e^{(i)} \in E^t$, we would expect $\rho^{(i)}$ to increase if $J(a^{(i)}, s_t) = 1$ and decrease if $J(a^{(i)}, s_t) = 0$. Recall from Section 3.3.1 that we model the unreliability $\rho^{(i)}$ as the expected value of a beta distribution $Beta_x[\alpha^{(i)}, \beta^{(i)}]$. Given that $\rho^{(i)} \in [0, 1]$, we can use $I(a^{(i)}, s_j)\ \forall e^{(i)} \in E^t$ and $\forall j = 0, ..., t$ as data points with which we can fit a beta distribution. For the sake of brevity, we drop the superscript (i) denoting the expert $e^{(i)}$, as the process is identical for all experts.

To fit the distribution, we employ the method of moments estimators, which operates by equating the moments of the beta distribution to the sample mean and variance and then solving for the parameters [Owen 2008]. After employing this method, we arrive at the following expressions for $\hat{\alpha}$ and $\hat{\beta}$

$$\begin{aligned} \hat{\alpha} &= \bar{X}\left(\frac{\bar{X}(1 - \bar{X})}{S^2} - 1\right) \\ \hat{\beta} &= (1 - \bar{X})\left(\frac{\bar{X}(1 - \bar{X})}{S^2} - 1\right), \end{aligned} \tag{3.10}$$

where $\bar{X}$ denotes the sample mean and S^2 the sample variance. The new unreliability estimate is then set to the expected value of the distribution $Beta_x[\hat{\alpha}, \hat{\beta}]$, according to Equation 3.4.

Chapter 4

Research Plan

4.1 Experiments

In order to test the hypotheses presented in Section 3.2.2, we devise a number of experiments on the medical diagnosis domain discussed in Section 2.2.2. Representing the domain as an influence diagram, we use the forward sampling algorithm to generate observations for the agent [Koller and Friedman 2009]. Each expert has access to the *ground truth* influence diagram used to sample observations. When giving advice, if the expert is reliable, they merely offer whichever action maximises the utility for the given state. If the expert is unreliable, we propose modelling the expert's unreliability as a beta distribution $Beta_x[\alpha_{true}, \beta_{true}]$. Then, with probability $\rho_{true} = \mathbb{E}[x]$, the expert gives the correct advice, and with probability $1 - \rho_{true}$, the expert chooses out of the remaining, suboptimal actions with uniform probability.

To evaluate the performance of our agent, we examine the rate of convergence to a policy by plotting the regret over the number of trials, averaged over a number of runs to smooth out sample variance, where the regret is given by

$$Regret_t = r(s_t, \underset{a}{\operatorname{argmax}}\ EU(a|s_t)) - r_t, \tag{4.1}$$

where $r(s_t, \underset{a}{\operatorname{argmax}}\ EU(a|s_t))$ is the reward for the optimal action given s_t and r_t is the reward received by the agent at trial t. Here the steepness of the curve indicates how quickly the agent has converged to a policy, with a steeper curve indicating faster convergence. If the average regret converges to 0, the agent has discovered the optimal policy. Otherwise, it has discovered a suboptimal policy.

In the first set of experiments, we have the agent learning with the advice of a reliable expert. This agent, that learns according to the process outlined in Section 3.3, is hence referred to as the *sceptical* agent. We compare the average regret over time of this agent against an ϵ-greedy agent learning without expert advice (hence referred to as the *default* agent) and against an agent that always follows the advice it is given (hence referred to as the *gullible* agent). We also examine the effect of changing the initial parameters α and β of the unreliability estimate for the *sceptical agent* as well as the tolerance parameters μ and γ for the expert.

In the second set of experiments, we introduce experts with varying degrees of unreliability. We compare the average regret over time of the *default*, *gullible* and *sceptical* agents. As the experts can now contradict each other, we have the *gullible* agent choose randomly between contradictory advice for a given state. Once again, we examine the effect of changing initial $\alpha^{(i)}$ and $\beta^{(i)}$, and the tolerance parameters μ and γ.

4.2 Time Plan

Having outlined our proposed experiments, we now divide our research methodology and experiments into a number of deliverables to be completed over the course of the research. These deliverables are as follows.

- Implementing the medical diagnosis influence diagram environment discussed in Section 2.2.2, as well as other single-stage decision problem influence diagrams, with varying degrees of domain complexity.
- Implementing the variable elimination algorithm to calculate the optimal policy for a given influence diagram, as outlined by Zhang and Poole [1994].
- Implementing the *default* ϵ-greedy agent as discussed in Section 2.1 and having it interact with the influence diagram environment.
- Implementing an expert, comprised of a *ground truth* influence diagram and the beta distribution that models unreliability.
- Implementing the structure and parameter learning algorithms discussed in Section 2.3, namely the parent lattice method by Buntine [1991] and the integer linear programming method by Bartlett and Cussens [2017].
- Implementing the unreliability model discussed in Section 3.3.1 and the update rule discussed in Section 3.3.3.
- Implementing the decision algorithm outlined in Section 3.3.2.
- Conducting the first set of experiments with completely reliable experts, as discussed in Section 4.1.
- Conducting the second set of experiments with unreliable experts, as discussed in Section 4.1.
- Running ablation studies to isolate critical components of the methodology and eliminating unnecessary constraints and assumptions.
- Interpreting results and, if necessary, conducting additional experiments.
- Writing the thesis document.

Dividing these deliverables across a 13 month time period from June 2020 to July 2021, we arrive at the following time plan.

Deliverable to be Implemented	**Time Period**		**Duration**
Influence diagram environments	07/06/2020	21/06/2020	2 weeks
Variable elimination for finding optimal policy	21/06/2020	05/07/2020	2 weeks
Default ϵ-greedy agent	05/07/2020	12/07/2020	1 week
Expert	12/07/2020	26/07/2020	2 weeks
Structure learning and parameter estimation algorithms	26/07/2020	18/10/2020	3 months
Unreliability model and update algorithm	18/10/2020	01/11/2020	2 weeks
Decision making algorithm	01/11/2020	15/11/2020	2 weeks
Reliable expert experiments	15/11/2020	13/12/2020	1 month
Unreliable experts experiments	13/12/2020	10/01/2020	1 months
Ablation studies	10/01/2020	07/03/2021	2 months
Additional tasks and result interpretation	07/03/2021	13/06/2021	3 months
Thesis write-up	13/06/2021	11/07/2021	1 month

References

[Bartlett and Cussens 2017] Mark Bartlett and James Cussens. Integer linear programming for the bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, 2017.

[Buntine 1991] Wray Buntine. Theory refinement on bayesian networks. In *Uncertainty Proceedings 1991*, pages 52–60. Elsevier, 1991.

[Burke and Klein 2020] Pierce Burke and Richard Klein. Confident in the crowd: Bayesian inference to improve data labelling in crowdsourcing. In *2020 International SAUPEC/RobMech/PRASA Conference*, pages 1–6. IEEE, 2020.

[Heckerman and Nathwani 1992] David E Heckerman and Bharat N Nathwani. An evaluation of the diagnostic accuracy of pathfinder. *Computers and Biomedical Research*, 25(1):56–74, 1992.

[Heckerman *et al.* 1995] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

[Howard and Matheson 2005] Ronald A Howard and James E Matheson. Influence diagrams. *Decision Analysis*, 2(3):127–143, 2005.

[Innes and Lascarides 2019] Craig Innes and Alex Lascarides. Learning structured decision problems with unawareness. In *International Conference on Machine Learning*, pages 2941–2950, 2019.

[Kao *et al.* 2018] Hao-Cheng Kao, Kai-Fu Tang, and Edward Y Chang. Context-aware symptom checking for disease diagnosis using hierarchical reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Knox and Stone 2009] W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16, 2009.

[Koller and Friedman 2009] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[Langford and Zhang 2007] John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 817–824. Citeseer, 2007.

[Lauritzen and Spiegelhalter 1988] Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988.

[Masegosa and Moral 2013] Andrés R Masegosa and Serafín Moral. An interactive approach for bayesian network learning using domain/expert knowledge. *International Journal of Approximate Reasoning*, 54(8):1168–1181, 2013.

[Murphy 2001] Kevin P Murphy. Active learning of causal bayes net structure. 2001.

[Ng *et al.* 2000] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[Norvig and Russell 1994] Peter Norvig and Stuart J. Russell. *Artificial Intelligence: A Modern Approach*. Prentice Hall, December 1994.

[Owen 2008] Claire Elayne Bangerter Owen. *Parameter estimation for the beta distribution*. Master's thesis, Brigham Young University-Provo, 2008.

[Pearl 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Sutton and Barto 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Torrey and Taylor 2013] Lisa Torrey and Matthew Taylor. 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, 2013.

[Yi *et al.* 2012] Sheng Kung Michael Yi, Mark Steyvers, Michael D Lee, and Matthew J Dry. The wisdom of the crowd in combinatorial problems. *Cognitive science*, 36(3):452–470, 2012.

[Zhang and Poole 1994] Nevin Lianwen Zhang and David Poole. A simple approach to bayesian network computations. In *Proceedings of the biennial conference-Canadian society for computational studies of intelligence*, pages 171–178. CANADIAN INFORMATION PROCESSING SOCIETY, 1994.