Learning Structured Decision Problems with Unawareness

Citation for published version:

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the 36th International Conference on Machine Learning (ICML)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Learning Structured Decision Problems with Unawareness

Craig Innes 1  Alex Lascarides 1

Abstract
Structured models of decision making often assume an agent is aware of all possible states and actions in advance. This assumption is sometimes untenable. In this paper, we learn influence diagrams from both domain exploration and expert assertions in a way which guarantees convergence to optimal behaviour, even when the agent starts unaware of actions or belief variables that are critical to success. Our experiments show that our agent learns optimal behaviour on small and large decision problems, and that allowing an agent to conserve information upon discovering new possibilities results in faster convergence.

1. Introduction
Probabilistic graphical models have proven useful in representing richly-structured decision tasks (Koller, 2009). Many techniques exist to learn a model’s structure and parameters from experience (Tsamardinos et al., 2006; Bartlett & Cussens, 2017) and expert advice (Masegosa & Moral, 2013). Unfortunately, all such methods assume the way the domain is conceptualized—the belief and action variables used to describe the problem—is completely known in advance. Often, this assumption does not hold.

In medicine, for example, suppose an agent prescribes a drug, but later a senior pharmacologist objects to the prescription based on a reason unforeseen by the agent—the patient carries a newly discovered genetic trait, and the drug produces harmful side effects in its carriers. Further, this discovery may occur after the agent has already learned a lot about how other (foreseen) factors impact the drug’s effectiveness. 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 hypotheses not yet considered. This example also shows that while it may be infeasible for an agent to gather all relevant factors of a problem before learning, it may be easy for an expert to offer contextually relevant advice during learning. Another example is in robotic skill learning. Methods like Cakmak & Thomaz (2012) teach a robot how to perform a new action it was unaware of, but not how to integrate it into an existing decision model.

Current learning and decision making models don’t address these issues; they assume the task is to use data to refine a distribution over a fixed hypothesis space. The above examples, however, illustrate a sort of reverse bayesianism (Karni & Viero, 2013), where the hypothesis space expands over time.

Rather than overcoming unawareness of states and actions, we could instead represent unawareness as an infinite number of hidden variables (Wood et al., 2006), or abandon learning a structured model and use densely connected hidden layers to implicitly learn structure from raw sensory input (Mnih et al., 2015). Both approaches have drawbacks. First, neither currently addresses how to adapt what one has learned when a new action is discovered. More importantly, the hidden layers/variables are not tied to grounded concepts with explicit meaning, so an agent cannot easily justify its decisions to a user, or articulate the limits of its current understanding to solicit help from others.

We instead propose an agent which explicitly overcomes its unawareness, and constructs an interpretable model of the problem. This paper makes three contributions: First, an algorithm which incrementally learns all components of a decision problem as an Influence Diagram (ID). This includes the reward function and probabilistic dependencies between variables, but also the set of belief and action variables themselves (Sections 2, 3.2). Second, an agent-expert communication protocol (Section 3.1) which interleaves contextual advice with learning, and guarantees our agent converges to optimal behaviour despite starting unaware of factors critical to success. Third, experiments on decision tasks of varying sizes showing our agent successfully learns optimal behaviour in practice (Section 4). Such contributions should be most relevant in interactive task learning (Laird et al., 2017), where agents must learn the underlying concepts which define a task.
Learning Structured Decision Problems with Unawareness

2. Learning with Full Awareness

We focus on learning observable, single-stage decision problems with discrete states and actions. In this paper, we learn optimal behaviour by learning an Influence Diagram (a Bayesian network extended with actions and rewards), which defines the optimal policy. We start with an agent who is aware of all states and actions, then address the task where the agent is unaware of relevant states and actions.

2.1. Making Optimal Decisions

Our agent’s goal is to learn the optimal policy \( \pi^* \) which, given evidence \( e \), chooses the action \( a \) which maximizes its expected utility \( \text{EU}(a|e) = \sum_{s'} P(s'|a,e)R(s') \) across all states \( s' \). If \( P(s'|a,e) \) or reward function \( R \) are unknown, the agent must learn them via trial and error. Our agent balances exploiting its current estimate \( \pi \) of \( \pi^* \) with exploring the domain by using an \( \epsilon \)-greedy policy (one which acts randomly with probability \( \epsilon > 0 \), and according to the current estimate of \( \pi \) with probability \( 1-\epsilon \)). For any policy \( \pi \), we can measure the expected loss in reward against the true optimal policy \( \pi^* \) using (1)—the policy error.

\[
PE(\pi) = \sum_e P(e) \left( \text{EU}(\pi(e)|e) - \text{EU}(\pi^*(e)|e) \right) \tag{1}
\]

If there are many states, evaluating the expected utility directly is infeasible. The next sections show how to exploit a problem’s inherent structure to make computations tractable.

2.2. Bayesian Networks

Bayesian networks (BNs) decompose the state space of the 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., \( s \in v(\mathcal{X}) \), where \( v(\mathcal{X}) \) is the cross-product of possible assignments to each variable in \( \mathcal{X} \)). An BN is a directed acyclic graph (DAG) \( Pa \) and parameters \( \theta \) which exploit the conditional independencies between variables in \( \mathcal{X} \). For each \( X \in \mathcal{X} \), \( Pa_X \subseteq Pa \) 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 \( \theta_X = P(X|Pa_X) \).

Given \( \mathcal{X} \), we can learn the most likely structure \( Pa^* \) and parameters \( \theta^* \) 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):

\[
P(Pa|D_{0:t}) \propto \prod_{X \in \mathcal{X}} \text{BD}_t(X, Pa_X) \tag{2}
\]

Here, \( \text{BD}_t(X, Pa_X) = P(Pa_X) \prod_{j \in v(Pa_X)} \frac{\beta(N_{0:j}^t + \alpha_{o|j}, \ldots, N_{m:j}^t + \alpha_{m|j})}{\beta(\alpha_{o|j}, \ldots, \alpha_{m|j})} \tag{3}
\]

Where the context is clear, we compress this notation to \( N_{o|j}^t \).

Unfortunately, evaluating the full posterior of (2), or even finding its maximum, is infeasible for even a moderate number of variables, as the number of possible BNs is hyper-exponential in the size of \( \mathcal{X} \) (Tian & He, 2009). To tackle this, most methods approximate \( Pa^* \) via either local search (Teyssier & Koller, 2005), sampling methods (Madigan et al., 1995), or by reducing the search space of structures considered reasonable (Buntine, 1991). For our task, we have another issue—most BN learning methods operate once on a single batch of data, and thus consider the collection of sufficient statistics (i.e., the BD-scores of each \( Pa_X \) and their associated \( N_{o|j} \)) to be a negligible pre-computed cost. In a decision problem, agents gather data incrementally, and exploit their current beliefs during learning, so we must update the sufficient statistics each time step.

To address these issues, we follow Buntine (1991) by constructing a reasonable parent lattice \( \mathcal{P}_X \) for each \( X \). Figure 1 shows an example lattice for \( X_1 \). Starting from \( \emptyset \), we construct larger parent sets by combining sets seen so far, and track the highest scoring set according to (3). Any parent-set with a score lower than some proportion \( \kappa \) of the best score, is considered “unreasonable”, and is not expanded further.

Once we have our reduced set \( \mathcal{P}_X \) for each \( X \), we can find the combination of reasonable parent sets which maximize (2) and form a valid DAG. In our task we will 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. Instead, following Bartlett & Cussens (2017), we treat finding \( Pa^* \) as the linear program given below:

\[\text{Maximize } \sum_{X \in \mathcal{X}} \text{BD}_t(X, Pa_X) \text{ subject to: } \]

\[\begin{align}
\sum_{Pa_X \in \mathcal{P}_X} P(Pa_X) &= 1 \\
\sum_{Pa_X \in \mathcal{P}_X} \text{BD}_t(X, Pa_X) &\leq \kappa \text{BD}_t(X, Pa^*_X)
\end{align}\]
Learning Structured Decision Problems with Unawareness

The variables \( I(Pa_X \rightarrow X) \) 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 \( \theta^* \) via (6):

\[
E_{\theta|D_{t+1}, Pa}(\theta_{ij}) = \frac{N_{ij} + \alpha_{ij}}{N_{i+} + \alpha_{i+}}
\]

(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) via (7).³

\[
BD_t(X, Pa_X) = BD_{t-1}(X, Pa_X) \frac{N_{ij} + \alpha_{ij} - 1}{N_{i+} + \alpha_{i+} - 1} \quad (7)
\]

Second, expressing structure learning as a linear program makes it easy to add extra structural constraints later (as we do in section 3.1). Third, each lattice \( P_X \) implicitly captures an approximate distribution over each \( X \)'s parents.⁴

In section 3.2, the agent uses this distribution to conserve information as its awareness of \( X \) expands.

2.3. Learning to Act with Influence Diagrams

An ID (Howard & Matheson, 2005) is a BN augmented with actions and rewards. It is a tuple \( \langle A, \mathcal{X}, \mathcal{R}, Pa, \theta \rangle \). 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 \( \mathcal{X} = \mathcal{B} \cup \mathcal{O} \), where \( \mathcal{B} \) is the set of variables the agent observes before taking action, and \( \mathcal{O} \) are the variables which describe the outcome of that action. The reward function \( R : v(\text{scope}(\mathcal{R})) \rightarrow \mathbb{R} \) gives the reward received for each state, where \( \text{scope}(\mathcal{R}) \subseteq \mathcal{X} \) are the variables which determine the reward. While \( Pa \) and \( \theta \) still describe the probabilistic dependencies, each \( Pa_X \) can now include belief and action variables (i.e., \( \forall X, Pa_X \subset \mathcal{X} \cup \mathcal{A} \)).

A domain trial at time \( t \) is a tuple \( d_t = \langle s_t, a_t, r_t \rangle \), where \( s_t = \langle b_t, o_t \rangle \), \( b_t \in v(\mathcal{B}) \), \( o_t \in v(\mathcal{O}) \), \( a_t \in v(\mathcal{A}) \) and \( r_t = R(s_t) \).

Figure 2 shows an example ID. Here, the agent chooses assignments for action variables TOPICAL CREAM and ORAL DRUG based on observing GEN X, then receives a reward based on outcomes DISEASE, and SIDE EFFECTS. Note that IDs often also include information arcs — edges from belief variables to action variables that show which variables the agent observes before making each decision. For simplicity of explanation, we omit information arcs, and assume in this paper that all action variables are set simultaneously, with access to observations of all variables in \( \mathcal{B} \).

We can use the same techniques from section 2.2 to learn \( Pa^* \) and \( \theta^* \) for IDs. Similarly, we can learn reward function \( R_t \) from \( \text{scope}(\mathcal{R}) \), and \( D_{0,j} \) by setting \( R_t(s_t|\text{scope}(\mathcal{R})) = r_t \), and by defaulting to indifference for unseen states (e.g., by setting the reward of unseen states to 0). Given a learned ID, the agent then chooses a policy which maximizes the expected utility from section 2.1 (where \( a \in v(\mathcal{A}) \) and \( e \in v(\mathcal{B}) \)).

3. Overcoming Unawareness

So far, we’ve assumed our agent was aware of all relevant variables in \( \mathcal{X} \cup \mathcal{A} \) and all members of \( \text{scope}(\mathcal{R}) \). 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 \( \text{scope}^+(\mathcal{R}) \), and the learner’s awareness of them at time \( t \) as \( \mathcal{X}^t, A^t \), and \( \text{scope}^t(\mathcal{R}) \).

³Here, \( s[\mathcal{Y}] \) projects assignment \( s \) over the variables in \( \mathcal{Y} \).
Suppose $X^+ = \{X_1, X_2\}$, $X^0 = \{X_1\}$, $A^+ = \{A_1, A_2\}$, $A^0 = \{A_1\}$. We assume the agent can’t observe the value of variables it is unaware of. In the medical example from section 1, if $X_2$ is a particular gene, we assume the agent cannot detect the presence of that gene if it is unaware it exists. 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). This means at time $t = 0$, the agent does not observe the true trial $d_0$, but rather $\langle s_0[A^0], a_0[A^0], r_0 \rangle$. But awareness of those missing factors may be crucial to learning $\pi_+$. For example, the best action may depend upon $v(X_2)$, or the optimal policy may sometimes involve setting $A_2$ to true. The next sections thus answer two main questions. First, how can an agent discover and overcome its own unawareness by asking for help? 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?

### 3.1. 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 paper assumes the expert is cooperative and infallible. Further, we abstract away 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., (Zettelmoyer & Collins, 2007) for work on this problem). We do not, however, assume the expert knows the agent’s beliefs about the TD. The variables $X^c$ and $A^c$ express the expert’s knowledge of our agent’s awareness, and include only those actions the expert has explicitly seen the agent perform, or variables the agent mentions in conversation.

As argued in the introduction, the goal is to provide a minimal set of communicative acts so that interaction between the agent and expert mirrors 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. The next sections identify four advice types, whose combination guarantee the agent eventually behaves optimally, regardless of initial awareness.

#### 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 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 (8-12) hold:

$$
\sum_{t' \leq t} \frac{t - t'}{\mu} > \beta \tag{8}
$$

$$
\sum_{t' \leq t} \frac{t - t'}{\mu} > \beta \tag{9}
$$

$$
EU(\pi_+(b_1) | b_1) - EU(\pi_+(b_1) | b_1) > \beta \tag{10}
$$

$$
\exists A', a' \in v(\mathcal{A} \cup A'), EU(a' | b_1) > EU(a_i | b_1) \tag{11}
$$

$$
\exists A'', a'' \in v(\mathcal{A} \cup A''), EU(a'' | b_1) > EU(a_i | b_1) \tag{12}
$$

Then the expert will utter advice of the form (13):

$$
EU(a' | w_i^a) > EU(a_i | \mathcal{A} \cup A') | w_i^a \tag{13}
$$

Equation (8) ensures some minimum time $\mu$ has passed since the expert last gave advice, while (9) ensures the expert won’t interrupt unless its estimate of the agent’s policy error is above some threshold $\beta$. Together, $\mu$ and $\beta$ define how tolerant the expert will be to the agent’s poor performance before deciding to give advice. Equation (10) ensures that the expert’s suggested action has an expected reward higher than what the agent received at time $t$. Finally, (11-12) ensure not only that a better action $a'$ exists, but also that $a'$ introduces the minimum amount of potentially unforeseen action variables $A'$ 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 (13)—the expert’s utterance—uses an ambiguous term $w_i^a$, whose intended meaning is the true state $b$ (i.e $[w_i^a] \in v(B^+)$), but whose default interpretation by the agent is $b[B^+]$. In words, the expert says “In the last step, $a'$ would have been better than $a_i$”.

By introducing ambiguity, the agent can now interpret (13) in two ways. The first is as a partial description of the true
Learning Structured Decision Problems with Unawareness

<table>
<thead>
<tr>
<th>∀A ∈ A′, A ∈ A⁺ ∧ ∃X ∈ scope⁺(R), anc(A, X) (14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>∀s ∈ v(ailer⁺), s[B™] = s₁[B™] ∧ R⁺(s) &gt; r₁ (15)</td>
</tr>
</tbody>
</table>

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 ID). Equations (14-15) 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⁺(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 (14) when learning Pa⁺ by adding the constraint (16) to our linear program:

| ∀Y ∈ scope(R) : ∑ \_{X ∈ A} I(Pa_X → X) ≥ 1 (16) |

The second way the agent could use (13) is by adding its default interpretation of the advice to its current knowledge:

| ∀b ∈ B⁺ : b[B™] = b₁[B™] ⇒ EU(a’|b) > EU(a|b) (17) |

The agent can then enforce (17) by choosing a’ whenever b[B™] = b₁[B™], regardless of what seems likely from D₀₋₁. 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.⁷

Lemma 1. Consider an agent with awareness A’ ⊆ A⁺, and expert following (8-13). As k → ∞, either PE(π⁺ₖ) → c ≤ β or the expert utters (13) where A’ ≠ ∅.

3.1.2. Resolving A Misunderstanding

We noted before that the agent’s default interpretation of (13) could lead it to misunderstand the expert’s intended meaning. To illustrate, suppose the agent receives advice (18) and (19) at times t - k and t:

| EU(a|wᵀ₋ₖ) > EU(a'|wᵀ₋ₖ) (18) |
| EU(a|wᵀ) < EU(a'|wᵀ) (19) |

While the intended meaning of each statement is true, the agent’s default interpretations of wᵀ₋ₖ and wᵀ may be identical. That is, bᵀ₋ₖ[B™] = b₁[B™]. From the agent’s perspective, (18) and (19) conflict, and thus give the agent a clue that its current awareness of X⁺ is deficient. To resolve this conflict, the agent asks (20) (in words, “which B has distinct values in b₁₋ₖ and b₁?”) and receives an answer which provides monotonic information of the form (21):

| ∀B ∈ B⁺ ∧ s₁₋ₖ[B™] ≠ s₁[B™] (20) |
| B’ ∈ B⁺ ∧ ∃X ∈ scope⁺(R), anc(B’, X) (21) |

Notice there may be many variables in B⁺ \ B’ whose assignments differ in b₁₋ₖ and b₁. Thus, the expert’s answer can be non-exhaustive. This means the agent must abandon its previous defeasible interpretations of the form (17), but can keep (14-15), as these are true regardless of what the true ID is. Lemma 2 guarantees the expert will reveal new belief variables provided misunderstandings can still arise.

Lemma 2. Consider an agent with awareness X’ ⊆ X⁺, A’ = A⁺. If ∃b’, b ≠ b₁[B™] = b₁[B™] and π⁺(b) ≠ π⁺(b’), then as k → ∞, either PE(π⁺ₖ) → c ≤ β, or the expert utters (21) such that B’ = B’.

3.1.3. Unforeseen Rewards

In typical IDs (where we assume the agent starts fully aware of X⁺, A⁺, and scope⁺(R)), we tend only to think of the trials as providing counts. For an unaware agent, a trial dₖ = ⟨sₖ, aₖ, rₖ⟩ also encodes monotonic information:

| ∀s, s[X’™] = s₁[X’™] ∧ R⁺(s) = r₁ (22) |

This, along with (15), constrain the form of R the agent can learn. Recall that scope⁺(R) may be only a subset of scope⁺(R), so it might be impossible to construct an R : v(scope⁺(R)) → R satisfying all descriptions (22) gathered so far. Further, those extra variables in scope⁺(R) \ scope⁺(R) may not be in X’™. To resolve this, if the agent fails to construct a valid R, it asks (23) (in words, “which variable X (that I don’t already know) is in scope⁺(R)?”), receiving an answer of the form (24):

| ∀X ∈ scope⁺(R) \ scope⁺(R) ∩ X’™ (23) |
| X’’ ∈ scope⁺(R) ∧ X’’ ∈ X⁺ (24) |
Again (24) may be non-exhaustive. Even so, we can guarantee that the agent’s reward function eventually equals $R_+$. **Lemma 3.** Consider an $\epsilon$-greedy agent with awareness $A' = A^+, \text{scope} (R) \subseteq \text{scope} (\mathcal{R})$. As $k \to \infty$, there exists a $K$ such that $\forall s, \forall k \geq K, R_{t+k} (s) = R_+ (s)$.

3.1.4. Unknown Effect

Recall that, to keep the problem tractable, our agent searches for IDs in the space of “reasonable” parent sets $P$. Unfortunately, there might be no valid ID within $P$ which also satisfies the constraints of the form (16). The most obvious case of this is when a variable $X \notin \text{scope}_i (\mathcal{R})$ has no children in any reasonable DAG (i.e., $\forall Y, \forall Pa_Y \in P_Y, X \notin Pa_Y$). Here, the agent can ask why $X$ is relevant by asking (25) (“what $V$ does $X$ affect directly?”). The answer (e.g., $X \in Pa_V$) informs past scores (26) to the agent:

$$\begin{align*}
?AV (X \in Pa_V) & \quad (25) \\
V' \in X^+ \land \exists Y \in \text{scope}_+ (\mathcal{R}), \text{anc} (V', Y) & \quad (26)
\end{align*}$$

3.2. Adapting to Unforeseen Factors

Section 3.1 showed four ways to expand the agent’s awareness of $A^*, A$, and $\text{scope} (\mathcal{R})$. To improve on simply restarting learning, we must now say how the agent adapts its beliefs about $Pa$ and $\theta$ when its awareness expands.

3.2.1. Adding a New Belief Variable

When the agent discovers a belief variable $Z$, its old distributions over parent sets no longer cover all possible parents. That is, for $X \in X^{t-1}$, $P_X$ did not cover cases where $Z$ was a parent of $X$. Worse, the agent cannot observe $Z$’s values in $D_0:t$, so cannot observe $N_{Z=0}^{t-k}$ or $N_{X=1}^{t-k}$ when $Z \in Pa_X$. The $\alpha$-parameters involving $Z$ are also undefined, yet we need them to calculate $Pa^*$ and $\theta^*$ via (6-7).

Discovering a new variable makes the size of each (observed) state dynamic, in contrast to standard problems where they are static (e.g., $\{X_1 = 0, X_2 = 1\}$ becomes $(X_1 = 0, X_2 = 1, Z = ?)$). We could phrase this as a missing data problem: $Z$ was hidden in the past but visible in future states, so estimate missing values and structure via e.g., expectation maximization (Friedman, 1998). But such methods commit us to costly passes over the full batch of data and makes finding $Pa^*$ for problems with more than a handful of variables intractable by eliminating the modularity of the BD-score. Alternatively, we could ignore states with missing information when we require counts involving $Z$. For example, we could use $P (Pa_X | D_{0:t})$ to score $Pa_X$ when $Z \in Pa_X$ but use $P (Pa_X | D_{0:n})$ when $Z \notin Pa_X$. But as Friedman & Goldszmidt (1997) point out, most structure scores, including (2), assume we evaluate models with the same data. If we compare models using different data sets (even two sets from the same distribution), the learner favours models evaluated with the smallest data set.

Instead, our method discards data $D_{0:t-1}$ gathered during the learner’s old view of the hypothesis space, but conserves the posterior probabilities learned from it to create new priors for $Pa$ and $\theta$ in the expanded space. On discovering the new variable $Z$, the agent updates each distribution over the parent sets $Pa_X \ (X \neq Z)$ to cover parent sets which include $Z$. Equation (27) creates a new prior $P'(Pa_X)$ from the old posterior. Here, $C$ is the normalizing constant and $0 < \gamma \leq 1$ expresses how much we trust our current $P$:

$$P'(Pa_X) = \gamma P'(Pa_X | Pa_X) + (1 - \gamma) P(Pa_X) \quad (27)$$

$$P'(Pa_X | Pa_X) =$$

$$\begin{cases} 
0 & \text{if } Pa_X \notin Pa_X \\
(1 - \gamma)BD_{t-1}(X, Pa_X) & \text{if } Z \notin Pa_X \\
\gamma BD_{t-1}(X, Pa'_X) & \text{if } Pa_X = Pa'_X \cup \{Z\}
\end{cases} \quad (28)$$

This update preserves the relative likelihood among the parent sets without $Z$. It also preserves our bias towards simple structures by giving only a small portion $\rho$ of the probability mass of parent sets without $Z$ to those with $Z$. This still leaves us to define a distribution over $Pa_X$ itself. The agent has no evidence on $Z$’s parents at the moment of $Z$’s discovery, so defaults to using the initial prior (4).

To adapt $\theta^*$, we return to the issue of the counts $N_{i,j}$ and $\alpha$-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 $N_{i,j}^{t-1}$, but keep the relative likelihoods they gave rise to by packing these into new $\alpha$-parameters, as shown in (29) ($K$ is a constant):

$$N_{i,j}^t = 0 \text{ for all } i, j, X, Pa_X$$

$$\alpha_{i,j} =$$

$$\begin{cases} 
\frac{K}{|Z|} P(j | \theta_{t-1}^*) & \text{if } X = Z \\
\frac{K}{Z} P(i, j | Pa_X \cup Z | \theta_{t-1}^*) & \text{if } Z \in Pa_X \\
K P(i, j | \theta_{t-1}^*) & \text{otherwise}
\end{cases} \quad (29)$$

Equation (29) summarizes the counts $D_{0:t-1}$ based on inferences from the old ID, then encodes these inferences in the $\alpha$-parameters of the new parameter prior. The new $\alpha$-parameters ensure that the likelihoods inferred from past states bias the estimate of $Pa^*$ and $\theta^*$ as more trials arrive. The larger $K$ is, the more what it learned before discovering $Z$ influences its reasoning after discovering $Z$.

3.2.2. 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
Algorithm 1 Learning IDs with Unawareness

1: Input: \( \mathcal{X}^0, \mathcal{X}^0, \mathcal{P}_0, \theta^0, \mathcal{P}^0 \)
2: for \( t = 1 \ldots \text{maxTrials} \) do
3: \( b_t \leftarrow \text{Generate new state} \)
4: \( \langle o_t, a_t, r_t \rangle \leftarrow \epsilon\text{-GREEDY}(b_t[\mathcal{X}^t]) \)
5: \( N_{ij}^t \leftarrow \text{Update with } b_t[\mathcal{X}^t], o_t[\mathcal{X}^t] \)
6: \( BD_t(X, Pa_X) \leftarrow \text{Update via (7) for } Pa_X \in \mathcal{P}_X \)
7: if \( t \equiv 0 \pmod{\tau} \) then
8: \( \text{Revise } \mathcal{P}^{t} \)
9: end if
10: \{ Unknown Effect (Section 3.1.4) \}
11: Ask (20), update \( \mathcal{X}^t, \mathcal{P}^t \) if any \( X \) has no children
12: \{ Unforeseen Rewards (Section 3.1.3) \}
13: \( \mathcal{R}^t \leftarrow \text{Update with constraints (22, 15)} \)
14: Ask (23), update scope_{\ell}(\mathcal{R}), \mathcal{X}^t, \mathcal{P}^t \) if update fails.
15: \{ Better Action Advice (Section 3.1.1) \}
16: if (8-12) are true then
17: \( \text{actAdvice} \leftarrow \text{Expert advises (13)} \)
18: Update \( \mathcal{X}^t, \mathcal{P}^t \) according to actAdvice
19: \{ Resolving a Misunderstanding (Section 3.1.2) \}
20: Ask (20) and update \( \mathcal{X}^t, \mathcal{P}^t \) when actAdvice conflicts with past utterances
21: end if
22: if \( \mathcal{X}^{t-1} \neq \mathcal{X}^t \) then
23: Update \( N_{ij}^t, \alpha_{ij}, \mathcal{P}, P(Pa) \) via (4, 27, 29)
24: end if
25: \( \langle Pa^t, \theta^t \rangle \leftarrow \text{Solve (5) with (16)}, \) and (6)
26: end for

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 \( D_{0:t-1} \).

3.2.3. 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 \( \text{scope}_{\ell}(\mathcal{R}) \) loosens the constraints of the form (16) which may allow us to construct a higher scoring ID structure that was previously invalid.

Algorithm 1 outlines the entire learning process. In most steps, the agent incrementally revises its model based on the latest trial and current reasonable parents. If enough time \( \tau \) passes, or the agent’s awareness expands, the agent makes larger changes to its model, including revising \( \mathcal{P} \). Given algorithm 1, theorem 1 guarantees our agent converges to a near-optimal policy (where “near-optimal” is bounded by the experts tolerance \( \beta \) ), regardless of initial awareness (provided \( \mathcal{P} \) includes the true structure).

Theorem 1. Consider an agent with initial awareness \( \mathcal{X}^0 \subseteq \mathcal{X}^+, A^0 \subseteq A^+, \text{scope}_{\ell}(\mathcal{R}) \subseteq \text{scope}_+(\mathcal{R}) \) following algorithm 1 without any pruning of the parent lattice (i.e., \( \kappa = 0 \)). As \( t \rightarrow \infty \), \( PE(\pi_t) \rightarrow c \leq \beta \).

4. Experiments

Our experiments show that agents following algorithm 1 converge to near-optimal behaviour not only in theory but also in practice. 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 three randomly generated IDs of increasing size: 12, 24, and 36 variables. Our results were similar across all sizes, but the differences between agents were most pronounced on the largest case, so we present those here (Full ID specifications and results for the small and medium cases are included in the technical supplement). In each, our agent begins with minimal awareness of the true ID \( (\mathcal{X}^0 = \{O_1\}, A^0 = \{A_1\}, \text{scope}_{\ell}(\mathcal{R}) = \{O_1\}) \). The agent acts in 5000 trials, using an \( \epsilon \)-greedy policy (\( \epsilon = 0.1 \)). We repeat experiments 100 times and average the results.

We use the cumulative reward across 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 \( R_t^\text{disc} = r_t + 0.99 \times R_{t-1}^\text{disc} \) for all \( t \).

We test several variants of our agent to show our approach is effective. The default agent follows algorithm 1 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 (4), (8), (9), (27), and (29). The nonConservative agent does not conserve information about \( \mathcal{P} \), \( Pa \), nor \( \theta \) via (27) and (29) when it discovers a new factor. Instead, it discards all trials and reverts to the original prior of (4). We include this agent to show the value of conserving information as \( \mathcal{X} \) and \( A \) expand. The non-relevant agent is like the default, but does not include any constraints of the form (16) when searching for \( Pa^* \). This means the agent might learn IDs where variables are completely disconnected. The truePolicy and random agents start knowing the true ID, and execute an \( \epsilon \)-greedy version of \( \pi_t \), or choose a random action respectively. These agents give an upper/lower bound on performance. The lowTolerance and highTolerance agents change the expert’s tolerance to \( \beta = 0.001 \) and \( \beta = 0.1 \).

4.1. Results

Across all IDs, our default agent converges to the optimal policy, despite starting unaware of factors critical to success. Figure 3a shows the cumulative reward of the default agent compared to the non-conservative, non-relevant, true-policy and random agents. The default agent converges to the op-
Learning Structured Decision Problems with Unawareness

Figure 3. Rewards (a and b) and size of $|A|$ (c) over time on the 36 variable task. Results for other tasks included in the supplement.

timal policy, and does so faster than its non-conservative counterpart. This shows conserving one’s beliefs on discovering new factors has value. 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 $\text{scope}^+(R)$ will be an ancestor to $\text{scope}^+(R)$ in $Pa^*$ regardless of whether we enforce (16). Actions the non-relevant agent leaves disconnected are those which exert little causal influence over $\text{scope}^+(R)$ (and thus usually have little 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 3b shows how the agent’s performance differs when varying the expert’s tolerance level: As the tolerance increases, the agent takes longer to converge towards a good policy, and also learns a worse final policy. Figure 3c shows why: Early on, the agent’s policy is “good enough” for the high tolerance expert, meaning the expert does not reveal the last few extra action variables which would net only small increases in the agent’s total reward.

5. Related Literature

Models of unawareness exist in logic (Board et al., 2011; Heifetz et al., 2013) and game theory (Feinberg, 2012), but interpret (un)-awareness from an omniscient perspective. We instead model awareness from the agent’s perspective and offer methods to overcome one’s unawareness.

Contextual bandits (Langford & Zhang, 2007) also deal with single-shot decision making as covered in this paper. Unlike our work, they focus on unstructured, noisy rewards, and don’t address unawareness of actions or states. Rong (2016) present Markov Decision Processes with Unawareness, which let an agent learn optimal behaviour even when unaware of some actions. We build on this work by providing a concrete mechanism for discovering unforeseen factors and by allowing the agent to discover explicit belief variables rather than atomic states—this is necessary to tackle large, richly-structured problems. McCallum & Ballard (1996) also learn an increasingly complex representation of the state space by gradually distinguishing between states which yield different rewards. Rather than dealing with unawareness, the focus of such approaches is on refining an existing state space. In other words, they do not support introducing unforeseen states or actions that the learner was unaware of before learning.

Several works use expert interventions to correct agent behaviour (Torrey & Taylor, 2013; Stone, 2009), or via agent-driven active learning (Masegosa & Moral, 2013; Murphy, 2001). Yet all such methods assume the expert’s intended meaning can be understood without expanding the agent’s conception of the state and action space. Our work allows experts to utter advice where ambiguity arises from their greater awareness of the problem.

6. Conclusion

We have presented an expert-assisted framework for learning structured decision problems, even when the learner starts unaware of factors critical to success. Further, our experiments show that being conservative about one’s beliefs improves the effectiveness of learning. In future work, we intend to extend our model to sequential tasks, as well as single-shot decision problems (Innes & Lascarides, 2019). Additionally, we aim to lift some assumptions imposed on the expert, and expand the range of situations in which the agent can ask for advice. For instance, we could let the expert be fallible (Masegosa & Moral, 2013), or leverage clues about the location of hidden variables present in the structure of the learner’s graphical model (Elidan et al., 2000) to guide questions asked to the expert.

Acknowledgements

This work is supported by EPSRC (UK). We would like to thank Kobi Gal, Subramanian Ramamoorthy, Benji Rosman,
Learning Structured Decision Problems with Unawareness

Stefano Albrecht, Vaishak Belle, and anonymous reviewers for their helpful feedback.

References


