Learning Structured Decision Problems with Unawareness

Anonymous Authors[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 bayesian decision networks 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) enable an expert to teach a robot *how* to perform a new action, but not *when* to use it. In lifelong learning scenarios, we want to integrate new skills into existing decision problems without forcing the robot to restart learning each time.

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. Under this framework, any change to the set of possible hypotheses constitutes an unrelated problem. The above examples, however, illustrate a sort of *reverse bayesianism* (Karni & Viero, 2013), where the hypothesis space itself 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 these approaches have drawbacks. First, neither currently address 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 a Bayesian Decision Network (BDN). 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).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

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 a *bayesian decision network* (a bayesian network extended with actions and rewards), which defines the optimal policy. We start with the scenario where the agent is aware of all possible 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 policy π^* which, given evidence e, chooses the action a which maximizes its *expected utility* $EU(a|e) = \sum_{s'} P(s'|a,e)\mathcal{R}(s')$ across all states s'. If $P(s|a,e)$ or reward function $\mathcal{R}$ are unknown, the agent must learn them via trial and error. Our agent balances *exploiting* its current estimate of π^* with *exploring* the domain by using an ϵ-*greedy policy* (one which acts randomly with probability $\epsilon > 0$, and according to the current estimate of π^* with probability $1-\epsilon$). For any policy π, we can measure the expected loss in reward against the true optimal policy π_+ using (1)—the *policy error*.

$$PE(\pi) = \sum_e P(e)\left(EU(\pi_+(e)|e) - EU(\pi(e)|e)\right) \quad (1)$$

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 $\mathcal{X} = \{X_1, \dots X_n\}$. Each state is s a joint assignment to all variables in $\mathcal{X}$ (i.e., $s \in v(\mathcal{X})$). An BN is a directed acyclic graph (DAG) Pa and parameters θ which exploit the conditional independencies between variables in $\mathcal{X}$. For each $X \in \mathcal{X}$, $Pa_X \in Pa$ is the *parent set* of X. Given Pa_X, the value of X is conditionally independent of all $Y \in \mathcal{X} \setminus Pa_X$ which are non-descendants of X. The parameters $\theta_X \in \theta$ then define the conditional probability distributions $\theta_X = P(X|Pa_X)$.

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

$$P(Pa|D_{0:t}) \propto \prod_{X \in \mathcal{X}} \mathrm{BD}_t(X, Pa_X) \quad (2)$$

$$\mathrm{BD}_t(X, Pa_X) = P(Pa_X) \prod_{j \in v(Pa_X)} \frac{\beta(N^t_{0|j} + \alpha_{0|j}, \dots, N^t_{m|j} + \alpha_{m|j})}{\beta(\alpha_{0|j}, \dots, \alpha_{m|j})} \quad (3)$$

Here, $\beta(n_1, \dots, n_m)$ is the *multivariate beta function*, and $N^t_{X=i|Pa_X=j}$ is the number of states in $D_{0:t}$ where X has assignment i and its parents have joint assignment j.[1] The $\alpha_{i|j}$ parameters come from the dirichlet priors over θ and act as a "pseudo-count" when data is sparse. We typically choose the prior $P(Pa)$ to penalize complex structures, as in equation (4) which assigns a cost of $\rho < 0.5$ for each extra parent:

$$P(Pa_X) = \rho^{|Pa_X|}(1-\rho)^{|\mathcal{X} \setminus Pa_X|} \quad (4)$$

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 $N_{i|j}$) 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 κ of the best score, is considered "unreasonable", and is not expanded further.[2]

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:

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

[2] In contrast to Buntine (1991) we also consider all subsets of a reasonable parent set to be reasonable.

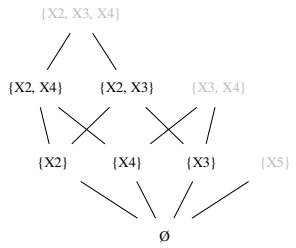


Figure 1. Parent lattice for X_1. Grey nodes are unreasonable

$$\begin{aligned} \max \quad & \sum_{X \in \mathcal{X}} \sum_{Pa_X \in \mathcal{P}_X} I(Pa_X \to X) \ln[BD(X, Pa_X)] \\ \text{s.t.} \quad & \sum_{Pa_X \in \mathcal{P}_X} I(Pa_X \to X) = 1 \qquad \forall X \\ & \sum_{\substack{X \in C \\ Pa_X \cap C = \emptyset}} I(Pa_X \to X) \geq 1 \qquad \forall C \subseteq \mathcal{X} \\ & I(Pa_X \to X) \in \{0, 1\} \quad \forall X, Pa_X \end{aligned} \tag{5}$$

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

Once we have learned the most likely structure Pa^*, we can learn its most likely parameters θ^* via (6):

$$\mathbb{E}_{\theta | D_{0:t}, Pa}(\theta_{i|j}) = \frac{N_{i|j} + \alpha_{i|j}}{N_{\cdot|j} + \alpha_{\cdot|j}} \tag{6}$$

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]

$$\text{BD}_t(X, Pa_X) = \text{BD}_{t-1}(X, Pa_X) \frac{N^t_{i|j} + \alpha_{i|j} - 1}{N^t_{\cdot|j} + \alpha_{\cdot|j} - 1} \tag{7}$$

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 $\mathcal{P}_X$ 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 $\mathcal{X}$ expands.

2.3. Learning to Act with Bayesian Decision Networks

A BDN (Russell & Norvig, 2002) is a BN augmented with actions and rewards. It is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{R}, Pa, \theta \rangle$. 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 $\mathcal{R} : v(scope(\mathcal{R})) \to \mathbb{R}$ gives the reward received for each state, where $scope(\mathcal{R}) \subseteq \mathcal{X}$ are the variables which determine the reward. While Pa and θ 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 $r_t = \mathcal{R}(s_t)$.

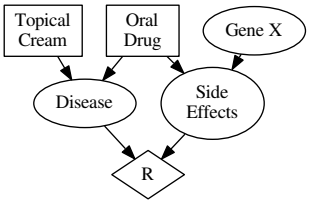


Figure 2. BDN for medical example. Rectangles are action variables, ovals are belief variables, and diamonds are reward nodes

Figure 2 shows an example BDN. Here, the agent chooses assignments for action variables TOPICAL CREAM and ORAL DRUG based on observing GENE X, then receives a reward based on outcomes DISEASE, and SIDE EFFECTS. We note that BDNs are essentially a subset of the notation used for *influence diagrams* (Howard & Matheson, 2005). Influence diagrams add *information arcs* from belief variables to action variables, which show the variables in $\mathcal{B}$ that the agent observes before choosing values for each $A \in \mathcal{A}$. For simplicity of explanation, we omit information arcs, and instead assume all action variables are set simultaneously, with access to observations of all variables in $\mathcal{B}$.

We can use the same techniques from section 2.2 to learn Pa^* and θ^* for BDNs. Similarly, we can learn reward function $\mathcal{R}_t$ from $scope(\mathcal{R})$, and $D_{0:t}$ by setting $\mathcal{R}_t(s_t[scope(\mathcal{R})]) = r_t$,[5] and by defaulting to indifference for unseen states (e.g., by setting the reward of unseen states to 0). Given a learned BDN, the agent then chooses a policy π^* 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 $scope(\mathcal{R})$. We now drop this assumption. From here onward we denote the true set of belief variables, actions, and reward scope as $\mathcal{X}^+$, A^+

[3] We include a proof of (7) in the technical supplement.

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

[5] Here, $s[\mathcal{Y}]$ projects assignment s over the variables in $\mathcal{Y}$.

and $scope_+(\mathcal{R})$, and the learner's awareness of them at time t as $\mathcal{X}^t$, A^t, and $scope_t(\mathcal{R})$.

Suppose $X^+ = \{X_1, X_2\}$, $X^0 = \{X_1\}$, $\mathcal{A}^+ = \{A_1, A_2\}$, $\mathcal{A}^t = \{A_1\}$. We assume the agent can't observe the value of variables it is unaware of. In the medical example from section 1, if X_2 is 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).[6] This means at time $t = 0$, the agent does not observe the true trial d_0, but rather $\langle s_0[\mathcal{X}^0], a_0[\mathcal{A}^0], r_0\rangle$. But awareness of those missing factors may be crucial to learning π_+. For example, the best action may depend upon whether X_2 is true, or the optimal policy may sometimes involve setting A_2 to a value other than its default. The 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., (Zettlemoyer & Collins, 2007) for work on this problem). We do not, however, assume the expert knows the *agent's* beliefs about the BDN. The variables $\mathcal{X}^e$ and $\mathcal{A}^e$ 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.

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

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:

$$t - t' > \mu \tag{8}$$

$$\sum_{t' \leq i \leq t} \frac{EU(\pi_+(b_i)|b_i) - EU(a_i|b_i)}{t - t'} > \beta \tag{9}$$

$$EU(a_t|b_t) \geq r_t \tag{10}$$

$$\exists \mathcal{A}', a' \in v(\mathcal{A}^e \cup \mathcal{A}'), EU(a'|b_t) > EU(a_t|b_t) \tag{11}$$

$$\nexists \mathcal{A}^{''}, (a'' \in v(\mathcal{A}^e \cup \mathcal{A}''), EU(a''|b_t) > E(a_t|b_t) \wedge |\mathcal{A}''| < |\mathcal{A}'|) \tag{12}$$

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

$$EU(a'|w_t^b) > EU(a_t[\mathcal{A}^t \cup \mathcal{A}']|w_t^b) \tag{13}$$

Equation (8) ensures some minimum time μ 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 β. Together, μ and β 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 $\mathcal{A}'$ needed to improve the agent's behaviour. This means there isn't another action a'' which could be advised while revealing fewer unaware variables to the agent. This follows from our desire to give non-exhaustive advice, by first suggesting improvements which use concepts the agent is already aware of, rather than introducing many new action variables all at once.

Equation (13)—the expert's utterance—requires explanation. On first thought, the expert should utter $EU(a'|b_t) > EU(a_t|b_t)$, thus fully describing b_t. But remember that

the agent's awareness $\mathcal{B}^t$ may be a tiny subset of $\mathcal{B}^+$. Uttering such advice may involve revealing many variables the agent is currently unaware of. This is exactly the type of cognitively-taxing exhaustive explanation we wish to avoid. Conversely, restricting the description to $\mathcal{X}^e$ could make the advice false: there may exist a $b' \neq b_t$ where $b'[\mathcal{X}^e] = b_t[\mathcal{X}^e]$, but $EU(a_t|b') > EU(a'|b')$. The answer is to use an *ambiguous term* w^b, whose *intended* meaning is the true state b (i.e $[\![w^b]\!] \in v(\mathcal{B}^+)$), but whose *default* interpretation by the agent is $b[\mathcal{B}^t]$. In words, the expert says *"In the last step, a' would have been better than a_t"*.

By introducing ambiguity, the agent can now interpret (13) in two ways. The first is as a *partial description* of the true BDN, which is true *regardless* of what it learns in future. On hearing (13), the agent adds (14-15) to its knowledge:

$$\forall A \in \mathcal{A}', A \in \mathcal{A}^+ \wedge \exists X \in scope_+(\mathcal{R}), anc(A, X) \quad (14)$$

$$\exists s \in v(\mathcal{X}^+), s[\mathcal{B}^t] = s_t[\mathcal{B}^t] \wedge \mathcal{R}_+(s) > r_t \quad (15)$$

Where $anc(X, Y)$ means X is the ancestor of Y (i.e., there is a directed path from X to Y in the true BDN). 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 $scope_+(\mathcal{R})$, and that there exists some state the agent could have reached which would yield a better reward than the one it got. In fact, we can enforce all equations of the form (14) when learning Pa^* by adding the constraint (16) to our linear program:

$$\forall Y \notin scope(\mathcal{R}) : \sum_{\substack{X \in \mathcal{X} \\ Pa_X : Y \in Pa_X}} I(Pa_X \to X) \geq 1 \quad (16)$$

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

$$\forall b \in \mathcal{B}^+ : b[\mathcal{B}^t] = b_t[\mathcal{B}^t] \Rightarrow EU(a'|b) > EU(a|b) \quad (17)$$

The agent can then enforce (17) by choosing a' whenever $b[\mathcal{B}^t] = b_t[\mathcal{B}^t]$, regardless of what seems likely from $D_{0:t}$. We should now see that even with a cooperative, infallible expert, even abstracting away issues of grounding natural language, misunderstandings can still happen due to differences in agent and expert awareness. As the next section shows, such misunderstandings can reveal gaps in the agent's awareness and help to articulate queries whose answers guarantee the agent expands its awareness.

Lemma 1 guarantees the expert's advice strategy continues to reveal unforeseen actions to the agent so long as its performance in trials exceeds the expert's tolerance.[7]

[7]We include proofs of all lemmas in the technical supplement

Lemma 1. *Consider an agent with awareness $A^t \subset A^+$, and expert following (8-13). As $k \to \infty$, either $PE(\pi_{t+k}) \to c \leq \beta$ or the expert utters (13) where $\mathcal{A}' \neq \emptyset$.*

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:

$$EU(a|w^b_{t-k}) > EU(a'|w^b_{t-k}) \quad (18)$$

$$EU(a|w^b_t) < EU(a'|w^b_t) \quad (19)$$

While the intended meaning of each statement is true, the agent's default interpretations of w^s_{t-k} and w^s_t may be identical. That is, $b_{t-k}[\mathcal{B}^t] = b_t[\mathcal{B}^t]$. From the agent's perspective, (18) and (19) conflict, and thus give the agent a clue that its current awareness of $\mathcal{X}^+$ is deficient. To resolve this conflict, the agent asks (20) (in words, *"which B has distinct values in b_{t-k} and b_t?"*) and receives an answer which provides monotonic information of the form (21):

$$?\lambda B(B \in \mathcal{B}^+ \wedge s_{t-k}[B] \neq s_t[B]) \quad (20)$$

$$B' \in \mathcal{B}^+ \wedge \exists X \in scope_+(\mathcal{R}), anc(B', X) \quad (21)$$

Notice there may be many variables in $\mathcal{B}^+ \setminus \mathcal{B}^t$ whose assignments differ in b_{t-k} and b_t. Thus, the expert's answer can be *non-exhaustive*. This means the agent must abandon its previous defeasible interpretations of the form (17), but can keep (14-15), as these are true regardless of what the true BDN is. Lemma 2 guarantees the expert will reveal new belief variables provided misunderstandings can still arise.

Lemma 2. *Consider an agent with awareness $\mathcal{X}^t \subset \mathcal{X}^+$, $\mathcal{A}^t = \mathcal{A}^+$. If $\exists b', \exists b \neq b', b[\mathcal{B}^t] = b'[\mathcal{B}^t]$ and $\pi_+(b) \neq \pi_+(b')$, then as $k \to \infty$, either $PE(\pi_{t+k}) \to c \leq \beta$, or the expert utters (21) such that $B' \notin \mathcal{B}^t$*

3.1.3. Unforeseen Rewards

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

$$\exists s, s[\mathcal{X}^t] = s_t[\mathcal{X}^t] \wedge \mathcal{R}_+(s) = r_t \quad (22)$$

This, along with (15), constrain the form of $\mathcal{R}$ the agent can learn. Recall that $scope_t(\mathcal{R})$ may be only a subset of $scope_+(\mathcal{R})$, so it might be impossible to construct an $\mathcal{R} : v(scope_t(\mathcal{R})) \to \mathbb{R}$ satisfying all descriptions

(22) gathered so far. Further, those extra variables in $scope_+(\mathcal{R}) \setminus scope_t(\mathcal{R})$ may not be in $\mathcal{X}^t$. To resolve this, if the agent fails to construct a valid $\mathcal{R}$, it asks (23) (in words, *"which variable X (that I don't already know) is in $scope(\mathcal{R})$?"*), receiving an answer of the form (24):

$$?\lambda X(X \in scope_+(\mathcal{R}) \bigwedge_{X' \in scope_t(\mathcal{R})} X \neq X') \tag{23}$$

$$X'' \in scope_+(\mathcal{R}) \wedge X'' \in \mathcal{X}^+ \tag{24}$$

Again (24) may be *non-exhaustive*. Even so, we can guarantee that the agent's reward function eventually equals $\mathcal{R}_+$.

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

3.1.4. Unknown effect

Recall that, to keep the problem tractable, our agent searches for BDNs in the space of "reasonable" parent sets $\mathcal{P}$. Unfortunately, there might be no valid BDN within $\mathcal{P}$ which also satisfies the constraints of the form (16). The most obvious case of this is when a variable $X \notin scope_t(\mathcal{R})$ has no children in *any* reasonable DAG (i.e., $\forall Y, \forall Pa_Y \in \mathcal{P}_Y, X \notin Pa_Y$). Here, the agent can ask *why* X is relevant by asking (25) (*"what V does X affect directly?"*). The answer (e.g., $X \in Pa_{V'}$) imparts information (26) to the agent:

$$?\lambda V(X \in Pa_V) \tag{25}$$

$$V' \in \mathcal{X}^+ \wedge \exists Y \in scope_+(\mathcal{R}), anc(V', Y) \tag{26}$$

3.2. Adapting to Unforseen Factors

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

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 $X \in \mathcal{X}^{t-1}$, $\mathcal{P}_X$ 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 $N^{t-k}_{Z=i|j}$, or $N^{t-k}_{X=i|j}$ when $Z \in Pa_X$. The α-parameters involving Z are also undefined, yet we need them to calculate Pa^* and θ^* 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., $\langle X_1 = 0, X_2 = 1 \rangle$ becomes $\langle X_1 = 0, X_2 = 1, Z = ? \rangle$) . 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 $P(Pa_X|D_{t:n})$ 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 θ 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 $\mathcal{P}$:

$$P'(Pa_X) = \gamma P'(Pa_X|\mathcal{P}_X) + (1-\gamma)P(Pa_X) \tag{27}$$

$$P'(Pa_X|\mathcal{P}_X) = \begin{cases} 0 & \text{if } Pa_X \notin \mathcal{P}_X \\ \frac{(1-\rho)}{C}\mathrm{BD}_{t-1}(X, Pa_X) & \text{if } Z \notin Pa_X \\ \frac{\rho}{C}\mathrm{BD}_{t-1}(X, Pa'_X) & \text{if } Pa_X = Pa'_X \cup \{Z\} \end{cases} \tag{28}$$

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 ρ of the probability mass of parent sets without Z to those with Z. This still leaves us to define a distribution over Pa_Z 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 θ^*, we return to the issue of the counts $N^t_{i|j}$ and α-parameters. As before, we wish to avoid the complexity of estimating Z's past values. Instead, we *throw away* the past states $D_{0:t-1}$ and their counts $N^{t-1}_{i|j}$, but *keep* the relative likelihoods they gave rise to by packing these into new α-parameters, as shown in (29) (K is a constant):

$$N^t_{i|j} = 0 \text{ for all } i, j, X, Pa_X$$

$$\alpha_{i|j} = \begin{cases} \frac{K}{|Z|}P(j|\theta^*_{t-1}) & \text{if } X = Z \\ \frac{K}{|Z|}P(i, j[Pa_X \setminus Z]|\theta^*_{t-1}) & \text{if } Z \in Pa_X \\ KP(i, j|\theta^*_{t-1}) & \text{otherwise} \end{cases} \tag{29}$$

Equation (29) summarizes the counts $D_{0:t-1}$ based on inferences from the old BDN, then encodes these inferences in the α-parameters of the new parameter prior. The new α-parameters ensure that the likelihoods inferred from past states bias the estimate of Pa^* and θ^* 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 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 $scope_t(\mathcal{R})$ loosens the constraints of the form (16) which may allow us to construct a higher scoring BDN 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 τ 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, regardless of initial awareness (provided $\mathcal{P}$ includes the true structure).

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

Algorithm 1 Learning BDNs with Unawareness

Input: $\mathcal{A}^0$, $\mathcal{X}^0$, Pa^0, θ^0, $\mathcal{P}^0$
for $t = 1 \ldots maxTrials$ **do**
  $b_t \leftarrow$ Generate new state
  $\langle o_t, a_t, r_t \rangle \leftarrow \epsilon$-GREEDY$(b_t[\mathcal{X}^t])$
  $N^t_{i|j} \leftarrow$ Update with $b_t[\mathcal{X}^t], o_t[\mathcal{X}^t]$
  $BD_t(X, Pa_X) \leftarrow$ Update via (7) for $Pa_X \in \mathcal{P}_X$
  if $t \equiv 0 \pmod{\tau}$ **then**
    Revise $\mathcal{P}^t$
  end if
  if Exists X with no reasonable children **then**
    Ask (25) and update $\mathcal{X}^t$, $\mathcal{P}^t$
  end if
  $\mathcal{R}^t \leftarrow$ Update with constraints (22, 15)
  if Update to $\mathcal{R}_t$ fails **then**
    Ask (20) and update $scope_t(\mathcal{R}), \mathcal{X}^t, \mathcal{P}^t$
  end if
  if (8-12) are true **then**
    $actAdvice \leftarrow$ Expert advises (13)
    Update $\mathcal{A}^t$, $\mathcal{P}^t$ according to $actAdvice$
    if $actAdvice$ conflicts with past utterances **then**
      Ask (20) and update $\mathcal{X}^t, \mathcal{P}^t$
    end if
  end if
  if $\mathcal{X}^{t-1} \neq \mathcal{X}^t$ **then**
    Update $N^t_{i|j}, \alpha_{i|j}, \mathcal{P}, P(Pa)$ via (4, 27, 29)
  end if
  $\langle Pa^t, \theta^t \rangle \leftarrow$ Solve (5) (with (16)), and (6)
end for

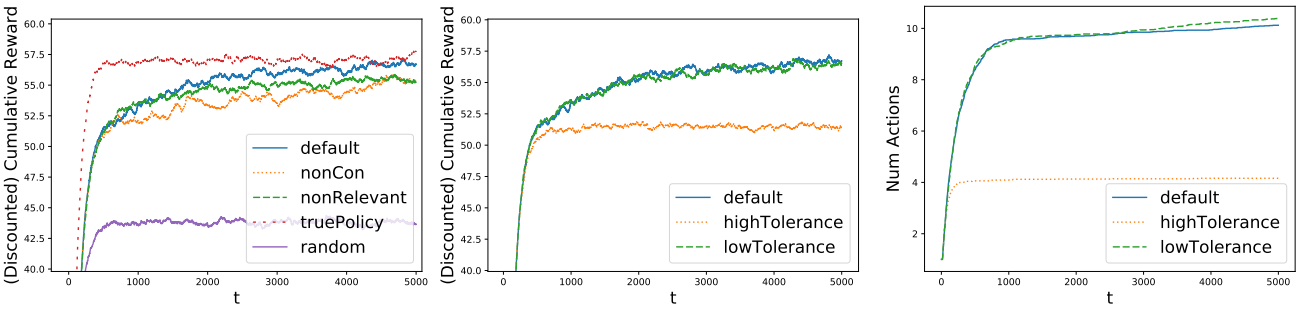
4. Experiments

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 BDNs 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 BDN 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 BDN ($\mathcal{X}^0 = \{O_1\}, \mathcal{A}^0 = \{A_1\}, scope_0(\mathcal{R}) = \{O_1\}$). The agent acts in 5000 trials, using an ϵ-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 $R^{disc}_t = r_t + 0.99 * R^{disc}_{t-1}$.

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 θ via (27) and (29) when it discovers a new factor. Instead, its 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 $\mathcal{A}$ 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 BDNs where variables are completely disconnected. The **truePolicy** and **random** agents start knowing the true BDN, and execute an ϵ-greedy version of π_+, or choose a random action re-



(a) (b) (c)

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

spectively. 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 BDNs, 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 optimal 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 $scope_+(\mathcal{R})$ will be an ancestor to $scope_+(\mathcal{R})$ in Pa^* regardless of whether we enforce (16). Actions the non-relevant agent leaves disconnected are those which exert little causal influence over $scope_+(\mathcal{R})$ (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.

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 (which 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 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 structural signatures (Elidan et al., 2000) to guide questions asked to the expert.

References

Bartlett, M. and Cussens, J. Integer Linear Programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, March 2017. ISSN 0004-3702. doi: 10.1016/j.artint.2015.03.003. URL http://www.sciencedirect.com/science/article/pii/S0004370215000417.

Board, O. J., Chung, K.-S., and Schipper, B. C. Two models of unawareness: Comparing the object-based and the subjective-state-space approaches. *Synthese*, 179(1):13–34, 2011.

Buntine, W. L. Theory Refinement on Bayesian Networks. *CoRR*, abs/1303.5709, 1991. URL http://arxiv.org/abs/1303.5709.

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

Coenen, A., Nelson, J. D., and Gureckis, T. M. Asking the right questions about human inquiry. *OpenCoenen, Anna, Jonathan D Nelson, and Todd M Gureckis."Asking the Right Questions About Human Inquiry". PsyArXiv*, 13, 2017.

Elidan, G., Lotner, N., Friedman, N., Koller, D., and others. Discovering hidden variables: A structure-based approach. In *NIPS*, volume 13, pp. 479–485, 2000. URL http://ai.stanford.edu/~nir/Papers/ELFK1.pdf.

Feinberg, Y. Games with unawareness. 2012.

Friedman, N. The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 129–138. Morgan Kaufmann Publishers Inc., 1998. URL http://dl.acm.org/citation.cfm?id=2074110.

Friedman, N. and Goldszmidt, M. Sequential update of Bayesian network structure. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pp. 165–174. Morgan Kaufmann Publishers Inc., 1997. URL http://dl.acm.org/citation.cfm?id=2074246.

Grice, H. P. Logic and conversation. *1975*, pp. 41–58, 1975.

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

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

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

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

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

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

Masegosa, A. R. and Moral, S. An interactive approach for Bayesian network learning using domain/expert knowledge. *International Journal of Approximate Reasoning*, 54(8):1168–1181, October 2013. ISSN 0888-613X. doi: 10.1016/j.ijar.2013.03.009. URL http://www.sciencedirect.com/science/article/pii/S0888613X13000698.

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

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

Murphy, K. P. Active learning of causal Bayes net structure. 2001. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.8206.

Rong, N. *Learning in the Presence of Unawareness*. PhD thesis, Cornell University, 2016.

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

Stone, W. B. K. a. P. Interactively Shaping Agents via Human Reinforcement: The TAMER Framework. 2009. URL http://www.cs.utexas.edu/users/ai-lab/?KCAP09-knox.

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

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

Torrey, L. and Taylor, M. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1053–1060. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

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

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