Learning Probabilistic Rules from Experience

Leslie Pack Kaelbling and Kurt Steinkraus
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

January 5, 2000

In this research note, we outline a research strategy for learning first-order world models. We have already begun the early steps of this work.

1 Probabilistic Propositional Models

Before we attempt to design algorithms for learning first-order models, it is important to understand the propositional case. In a propositional model, states of the world are described by *state variables* (also sometimes called *aspects* or *fluents*). The state variables in a factory control domain might be, for example, the temperature and pressure at different points in the factory.

It is popular to represent probabilistic propositional models of the world dynamics as *dynamic Bayesian networks* (DBNs). DBNs consist of two parts: a graph and a set of probability tables. The graph encodes the dependence relationships between the state variables at some generic time t and the state variables on the next time step, $t+1$. The tables encode the conditional probability distributions describing exactly how the value of a variable depends on its parents in the graph. (For more detail, see Dean and Kanazawa (1989).)

DBNs (and Bayesian networks, in general) have been successful representationally because they take advantage of the fact that in most large domains, not all variables are dependent on all others. By making independence an unstated default, DBNs often provide very compact representations of complex domains.

In this project, we are going to explore an alternative, rule-based representation language, inspired by the work of Drescher (1991). There are two reasons for this shift. First, we believe that rules will, in general, provide an even more compact representation of typical dynamic domains. Second, it seems more straightforward to to extend a rule-based representation to the first-order case.

Our rules will be of the form $P \xrightarrow{a,p} Q$, with approximate semantics: if P is true at time t, R is false at time t, and action a is taken, then R will be true at time $t+1$ with probability p. There is a major implicit default assumption here: If there is no rule to indicate otherwise, then state variables retain their values over time. This is the solution that the STRIPS (Fikes and Nilsson, 1971) representation system uses to the frame problem. In addition, we make a default independence assumption: given the rules $P \xrightarrow{a,p_1} Q$ and $R \xrightarrow{a,p_2} S$, we conclude by default that if P and R are true and Q and S are false at time t, then after doing action a, Q and S will be true with probability $p_1 \cdot p_2$. If this is not the case, then the rule set must contain another rule, with antecedent P, R that will override this default combination.

We have shown, though the details are omitted here, that rule sets of this kind describe well-defined joint probability distributions over the state variables at time $t + 1$ given the state variables at time t, and that any possible joint probability distribution can be represented in a rule set.

To give intution for the relationship between our rule sets and DBNs, we will work through a simple example. Consider the following rule set:

$$\begin{aligned} P, Q &\xrightarrow{a,.9} R \\ S, T &\xrightarrow{a,.6} \bar{U} \\ P &\xrightarrow{a,.7} R \end{aligned} \tag{1}$$

The corresponding DBN, including probability tables, is shown in figure 1 (The action is left implicit here; it would just be another node in the first layer, with arcs going into each node in the second layer). The tables are clearly inefficient at encoding the local conditional probability distributions. Using tree-structured tables (Boutilier et al., 1996), we get a more compact representation, as shown in figure 2. Nonetheless, for domains in which most things stay the same from time step to time step, the rule set is a much more economical representation.

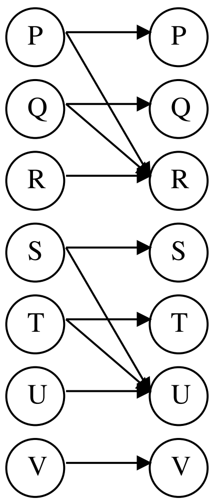
2 Learning Probabilistic Rules

The following is an adaptation of parts of Drescher's rule-learning algorithm. The algorithm proceeds in two major steps.

First, we look for situations in which an action seems to have an effect on some state variable. We will look for rules of the form $\xrightarrow{a,p} R$ by comparing empirical estimates of $\Pr(R_{t+1}|a_t, \bar{R}_t)$ with $\Pr(R_{t+1}|\bar{a}_t, \bar{R}_t)$. If the first is significantly greater, then we can conclude that R turns on more often as a result of doing action a than as a result of doing other actions, and will spawn the rule $\xrightarrow{a,\Pr(R_{t+1}|a_t,\bar{R}_t)} R$.

The next step is to strengthen existing rules by adding antecedents. For example, we might consider adding Q as an antecedent to the previously constructed rule. We would compare empirical estimates of $\Pr(R_{t+1}|a_t, \bar{R}_t, Q_t)$ with $\Pr(R_{t+1}|a_t, \bar{R}_t, \bar{Q}_t)$. If the first is significantly greater, then we will spawn the rule $Q \xrightarrow{a,\Pr(R_{t+1}|a_t,\bar{R}_t,Q_t)} R$, which is more reliable in predicting effects.

We will have to add considerably to this basic sketch to get a working algorithm. Important extensions include:



	P_{t+1}
P	1.0
$\bar{P}$	0.0

Tables for Q, S, and T are the same as for P.

	R_{t+1}
PQR	1.0
$PQ\bar{R}$	0.9
$P\bar{Q}R$	1.0
$P\bar{Q}\bar{R}$	0.7
$\bar{P}QR$	1.0
$\bar{P}Q\bar{R}$	0.0
$\bar{P}\bar{Q}R$	1.0
$\bar{P}\bar{Q}\bar{R}$	0.0

	U_{t+1}
STU	0.4
$ST\bar{U}$	0.0
$S\bar{T}U$	1.0
$S\bar{T}\bar{U}$	0.0
$\bar{S}TU$	1.0
$\bar{S}T\bar{U}$	0.0
$\bar{S}\bar{T}U$	1.0
$\bar{S}\bar{T}\bar{U}$	0.0

Figure 1: Dynamic Bayesian network corresponding to rule set 1.

.

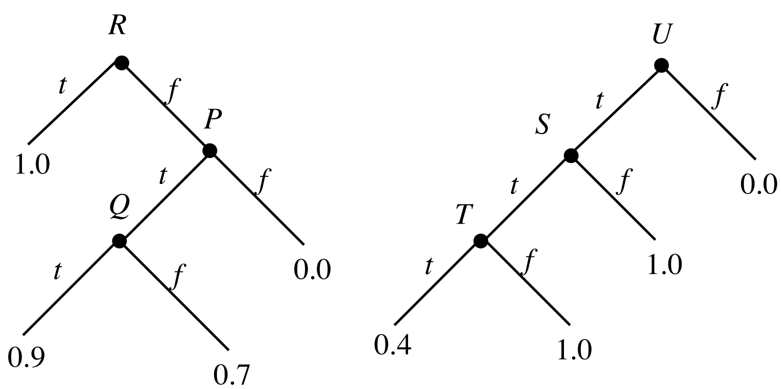


Figure 2: Compact representation of conditional probability tables from figure 1 using tree structures. The tree on the left encodes the conditional probability of R_{t+1} as a decision tree; the one on the right, U_{t+1}.

- Adding heuristics to limit which conjunctions of antecedents are considered. For instance, it may make sense to start by looking at state variables that represent information that is spatially "near". This will actually make more sense in the first-order case.
- Detecting dependencies and adding rules that override the default independence assumptions.
- Working in environments that are k-Markov, rather than completely observable, by considering as antecedents the values of state variables from previous time steps.

3 Limited First-Order Models

Our first ideas about working with limited first-order models make admittedly unrealistic assumptions about the perceptual system. In the course of this research, we expect to weaken these assumptions and, in particular, add an attentional mechanism to the vision system that will require the learning system to explicitly focus perception on "interesting" aspects of the current visual image.

A more naive perceptual model is that the vision system can deliver an existentially quantified statement of the form:

$$\exists xy.P(x) \wedge Q(y) \wedge R(x,y) \ .$$

That is, it describes a set of objects, their properties and relations. The statement may have an uncertainty attached; or the perception system might deliver a set of alternative hypotheses about what it its viewing.

We can learn a model in the form of universally quantified probabilistic rules that describe the general dynamics of the environment:

$$\forall xy.P(x) \wedge R(x,y) \xrightarrow{a,p} \neg Q(y) \ .$$

Such a rule, in combination with the perceptual statement described above, would allow us to infer, with probability p that at time $t+1$

$$\exists xy.P(x) \wedge \neg Q(y) \wedge R(x,y) \ .$$

The same notions of default independence and unchanging truth values over time will apply to these rule sets. Because of their very limited form, matching and forward inference will be very simple. We believe we can extend the algorithm for learning propositional rules fairly directly to this limited first-order case.

4 Extensions

A large number of important questions and extensions arise:

- The model described above does not preserve any notion of object identity across time steps. The x for which P holds at time t is not identified with the x for which it holds at $t+1$. We will consider the question of when and how object identity is important for planning, and try to develop ways of expressing it in the rule base. It will have to arise, logically, from some built-in knowledge about objects, such as the fact that if two very similar objects are seen at the same location (or slightly different locations) in two successive frames, then it's highly likely that those objects are identical.

- What is the role of indexical-functional (or deictic) representations? Agre and Chapman (1987) argue that it is useful to describe objects in descriptive terms, such as *the-block-that-is-in-front-of-me*. This seems reasonable, and we will investigate the use of such representations in this system.

- How can we parameterize actions? Once we can speak of objects, it may be appropriate to describe our actions in an object-centered way, such as $pickup(x)$. Indexical-functional names for objects might be especially useful here, so we could say things like *pickup*(*the-object-I-am-fixated-on*). By moving our visual fixation point, we implicitly change the parameterization of the action and therefore don't need to include the parameter explicitly.

- How can we handle partial observability? It is relatively easy to think about searching back through recent time steps to look for attributes on which our rules should be conditioned, at least in the propositional case. In the first-order case, it's harder, because of the lack of identification of objects across time steps. Drescher's "synthetic item" mechanism provides an interesting way to name new conditions in the world in terms of their connection to other conditions and actions we already know about. For instance, the system might decide that it's important to consider "the state of my car such that when I turn the key the starter works." It may be useful to name and work with that condition, trying to discover, for instance, which actions make it true in which circumstances.

References

Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, volume 1, pages 268–272, Seattle, Washington, 1987. Morgan Kaufmann.

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, San Francisco, California, 1996. Morgan Kaufmann.

Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3), 1989.

Gary L. Drescher. *Made-up Minds: A Constructivist Approach to Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts, 1991.

Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971. Reprinted in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate, eds., Morgan Kaufmann, 1990.