Learning in Worlds with Objects

Leslie Pack Kaelbling, Tim Oates, Natalia Hernandez and Sarah Finney
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Introduction

We are interested in building systems that learn to interact with complex real world environments, by representing the dynamics of the world with models that allow strong generalization through representation in terms of objects[1]. Humans speak (and apparently think) of the world as being made up of objects. There are chairs and apples and clouds and meetings. Certainly, part of the basis for this view is that there are clumps of coherent physical material that tend to be well-described in the aggregate. Even without engaging in the philosophical debate about whether objects *really* exist (Smith 1996), it is hard to imagine a truly intelligent agent that does not conceive of the world in terms of objects and their properties and relations to other objects.

For an agent to act effectively in our world, it must know (or, to make a weaker statement, act as if it knows) something of the form: *If object A is on object B, then if I move object B, object A will probably move too.* Such statements offer an ability to compactly express generalized information that cannot be approached without the description of the world in terms of objects.

Currently, we have no effective techniques for robust learning and reasoning about objects in uncertain domains. There are some theoretical foundations in probabilistic logic (Nilsson 1986; Halpern 1990), but they do not offer practical implementable methods. We can take inspiration from some relevant work in Bayesian networks (Koller & Pfeffer 1997; Getoor *et al.* 1999) and also in probabilistic inductive logic programming (Muggleton forthcoming), but we will have to develop additional new techniques to solve this problem. Our goal is to integrate representational ideas from classical AI with modern learning and uncertain reasoning methods, avoiding logic's problems of inferential intractability, perceptual ungroundedness, and inability to represent uncertainty.

[1]By "objects" we mean real things, like cups, tables, and possibly meetings, rather than the computational entities of object-oriented programming.

We wish to build an agent that is embedded in its environment. The agent can take actions that change the underlying state of the environment, and can make observations of the environment's state. In general, the observations will not reveal the true state of the environment: they will be noisy, and many underlying environmental states will look the same. The agent will also have a special scalar input signal, called reward, which is correlated with the underlying state of the environment. The agent's goal is to act in such a way as to gain a large amount of reward over time. It will have to learn about its environment in order to learn how to behave in a valuable way.

In all of this work, the goal is to learn to carry out some externally rewarded task, by taking actions in the environment based on perception. Although we will try to specify the learning task in a representation-independent way, we believe that the choice of internal representation and associated learning algorithms will have a huge effect on the efficiency, and even possibility, of the learning process.

Simple Example Domain

It is almost embarrassing to say this, but we are working in a simple blocks-world domain, in which there are multiple blocks, stacked in piles on a table. The blocks are made of different materials and are painted different colors.

We assume that the blocks are stacked in fairly orderly piles, so that each block is either on the table or on a single other block. The agent can perceive all of the blocks, their colors and materials, and their spatial relations. It can take action by picking up any clear block and putting it down on any other clear block or on the table.

Unlike the traditional blocks-world model, this world has substantial uncertainty. A real vision system will deliver perceptual descriptions of (parts of) the state of the world. As blocks are piled into higher towers, there will be an increasing probability that the tower is knocked over. Tall towers may be "buttressed" against collapse by placing other towers near them.

Within this general domain, the agent is given a task by rewarding it whenever it puts the blocks in front of it into the appropriate configuration. We use the term "task" to describe the kinds of block configurations for which the agent is rewarded and "initial state" to describe an initial configuration of objects. Possible tasks include:

- Pick up a green block.
- Cause every green block to have a red block on top of it.
- Make a copy of the stack of blocks that is on the far left of the table.
- Make a very high tower.
- Cause a very loud crash (by making a very high tower and then knocking it down).

There are endless small extensions that can make this domain much more interesting and complex. Blocks of different materials may be more or less slippery or deformable, changing the stability of the stacks. Blocks might come in different sizes; might open and close; and might be put inside other blocks.

Our first goal will be to learn general strategies for solving individual tasks from all (or many) initial configurations. Even within a single task, we can investigate "transfer" of knowledge to see whether learning to solve one initial state will allow the agent to solve other, similar, initial states more quickly. We are also interested in the more general question of how learning to solve a particular task can speed learning of other tasks in the same domain.

This kind of toy domain has been used in a huge amount of widely known planning research. It has also been used in some work on learning. Particularly relevant is work by (Whitehead & Ballard 1991) on partially-observable blocks worlds and by (Baum & Durdanovic 2000) and by (Dzeroski, de Raedt, & Blockeel 1998) on learning truly general block-stacking behaviors.

Perception and Action

Most work on reinforcement learning assumes that the agent has complete, perfect perception of its environment state. In the simple blocks world described above, that might not be implausible; but in any kind of realistic domain, it is. You can't see everything that is going on in your building; there are occlusions and distant objects. Furthermore, if you *could* see everything, you would still have to focus your attention on some small subset of the available information.

Ideas of attention, both physical attention due to moving gaze and implicit attention to areas within a fixed visual field, have been captured in work on active perception. The idea is that an agent makes observations of its environments through a fairly narrow channel, such as a visual fovea, and can take action to move the channel around. This converts the perceptual problem from one of dealing with a vast amount of data in parallel to a sequential control problem that explicitly selects the data required.

(Ullman 1984) characterized a basic set of visual primitives and described their aggregation into *visual routines.* (Chapman 1991) took these ideas and applied them in the visual system for a video-game player. We'll use a similar set of visual primitives in thinking about the blocks world.

One important idea is that of a *visual marker.* The agent is assumed to be able to keep track of a small number (say five) of objects in its visual field. It does so by "putting a marker" on an object. Having a marker on an object allows it to be the subject of visual queries, such as "What is the color of the marked object?" or "Is marked object 1 above marked object 2?" Markers can be placed on objects by relative motion: "move marker 2 to the top of the stack" or "move marker 3 one square to the left." Markers can also be placed on objects via *visual search.* For simple image properties, called "pop-out" properties, this search can be done very quickly in the human visual system (Triesman 1985). Thus, we can have as a visual primitive "move marker 3 to a red object" or "move marker 4 to a vertically-oriented object". More complex image properties require a slower search process. We may include them in our visual repertoire, but perhaps model their use with an additional time cost.

With just a small set of visual markers, the amount of information available in a single observation is fairly small; but by moving the markers around, it is possible, over time, to gain a fairly complete picture of the environment. This kind of partial observability requires the agent's strategy to have memory, allowing it to combine observations over time into an internal "view" of the current situation. It also requires a much more sophisticated set of learning techniques, which we discuss in more detail later.

What to Learn?

An agent embedded in a complex environment, trying to act so as to gain reward, can learn knowledge about its environment in different forms. There are trade-offs involved in learning the different forms of knowledge.

The most general thing an agent can learn is a model of the world dynamics. That is, a predictive model that maps a state of the world and an action of the agent into a new state of the world (or a probability distribution over new states of the world). Such a model can be used as the basis for planning (calculating which action to take given the current situation and a goal), and may be completely independent of the agent's particular reward function. Thus, knowledge gained about the general workings of the environment when learning to do one task may be directly applied to another task.

When the environment is completely observable, this learning problem is not (conceptually) too difficult. When it is partially observable, learning the world dynamics is equivalent to learning a complex finite-state automaton or hidden Markov model (HMM). In highly complex domains, however, the entire world model will simply be too complex for the agent to represent. It will have to learn *aspects* of the world dynamics that are relevant to achieving its goals. Thus simply learning world dynamics may be too un-focused a problem.

When the agent's reward function is fixed, it is not necessary to learn a model of the world dynamics. Although the world dynamics, coupled with the reward function, entail an optimal behavior for the agent, it may be possible to simply learn the policy directly. In a completely observable environment, a policy is a mapping from states of the world to actions (or a distribution over actions); in a partially observable environment, it is typically some kind of finite state machine that takes in observations and generates actions.

In general, it is hard to learn policies directly. In completely observable environments, it is much easier to learn a value function and derive a policy from it, as described below. In partially observable environments, it remains difficult to learn policies directly, but there are fewer good alternative strategies.

The one general-purpose policy-learning method is gradient descent. If the general structure of the policy is given as a parametrized model, in which the probabilities of taking actions are differentiable functions of the parameters, then the agent can adjust those parameters based on its experience in the world. These methods seem to work well for simple policy classes, but may have serious problems with local optima in larger policy classes.

An advantage of learning a policy is that, once learned, a policy may be executed very quickly, with no deliberation required to choose an action. A disadvantage is that there is usually no knowledge transfer. Having learned a policy for one task, it is not easy to use it to advantage in learning a policy for another task.

In completely observable environments, it is often advantageous to learn a *value function* rather than a policy. A value function maps states of the environment to a measure of the long-term reward available from that state (assuming the agent acts optimally thereafter). Given a value function, it is easy to compute the optimal policy, which is to take the action that leads to the highest expected value. Basic reinforcement-learning methods, such as Q-learning (Watkins & Dayan 1992), can effectively learn the value function from experience.

In partially observable environments, value functions are not defined over observations. One set of learning methods attempts to partially reconstruct the state of the environment based on the history of observations and then to apply standard reinforcement learning to this space. We find this technique to be particularly attractive and will discuss it further below.

Representational Strategies

We need to choose a method for representing the agent's observations, actions, and internal representations of states of the world; and to build on those representations to encode the knowledge that the agent learns about the world over time. The choice of representation may give opportunities for generalization (make learning easier in terms of number of examples required); make learning computationally easier or harder; or make planning (using the learned knowledge to choose actions) computationally easier or harder.

Propositional vs First-Order A state of the simple blocks world is a configuration of the blocks, including their colors, materials, and support relationships. Let us assume that individual blocks of the same material and color are completely interchangeable; there is no reason to distinguish particular individuals.[2]

The most naive representational strategy would assign an atom to each possible configuration. We could speak of configuration 5434, for example, leading to configuration 10334 after taking a particular action. This representation is simple in the sense that a single number can represent an entire state. Because it is completely unstructured, however, it doesn't give any leverage for generalization; there is no reasonable notion of two states being similar, based on this representation.

The next level of representational structure would be to encode a state of the world using a set of Boolean-valued variables corresponding to propositions that are true or false of the current configuration of the world. Example propositions would include: *block10-is-green*, *block2-is-plastic*, and *block3-is-on-block5*. A particular state of the world would be a vector of bits, one corresponding to each possible primitive proposition that could be true in the world.

Propositional representations afford much opportunity for generalization; the distance between states can be measured in terms of the number of bits that are different in their encodings. Learning algorithms such as neural networks and decision trees take advantage of propositional structure in the encoding of objects. It might be possible to learn, for instance, that any state in which all of the blocks are green has high reward, independent of the values of the stacking propositions. This knowledge could be stored much more compactly and learned from many fewer examples in the propositional rather than the atomic representation.

In the propositional representation, only part of the structure is revealed. That the propositions are derived from properties and relations among underlying objects is lost. In a relational representation, we retain the idea of objects. Thus, rather than having a fixed bit to stand for *block4-is-on-block2*, we would encode that fact relationally as *on(block4,block2)*. A state is typically encoded by listing the true (or known) propositions, but this time describing the propositions using their relational structure.

Such a representation affords even more opportunity for generalization. We might be able to learn and represent much more efficiently that, for example, states in which any block is on top of block 2 are rewarding.

[2]Note that this would not be true if, for instance, we could put things inside the boxes; then it might be of crucial importance which green plastic box contains my house key.

We can get further generalization if we allow quantification in our representation. So far, we have had to name the individual blocks and haven't been able to directly take advantage of the idea, described at the beginning of this section, of interchangeability. If we allow individual states to be described using existential quantification over the objects, then we don't have to name them. We can say

$$\exists x, y . red(x) \wedge blue(y) \wedge on(x, y)$$

to describe a situation in which a red block is on a blue block, generalizing over which particular red and blue blocks are on one another. Of course, general quantification leads to a very rich, but possibly inferentially difficult language.

We propose to investigate relational representations, with a particular emphasis on those that can be learned and used efficiently. Here is one example of a restricted but useful quantified relational description. If the agent's information state (either delivered directly from its perceptual system or resulting from an aggregation of percepts over time) can be represented as an existentially quantified statement of the form

$$\exists x, y . p(x) \wedge q(y) \wedge r(x, y)$$

and the world dynamics are described by universally quantified rules of the form

$$\forall x, y . p(x) \wedge r(x, y) \rightarrow next \;\; b(y)$$

then given an information state, predictions about the next information state will have the same existentially quantified form. Thus, the resulting state description can be the basis of the same type of predictive inference step with no additional complexity.

Deictic Representations From the perspective of opportunity to generalize, relational representations seem like the best choice. However, we have much more experience and facility with learning and with probabilistic representation in propositional representations. We may be able to have the best of both worlds by taking advantage of *deictic* or *indexical-functional* representations. Deictic representations may also help solve the problem of naming objects without necessarily identifying them as individuals (such as *block7*).

A deictic expression is one that "points" to something; its meaning is relative to the agent that uses or utters it and the context in which it is used or uttered. "Here," "now," and "the book that is in front of me" are examples of deictic expressions in natural language.

Two important classes of deictic representations are derived directly from perception and action relations between the agent and objects in the world. An agent with directed perception can sensibly speak of (or think about) *the-object-I-am-fixated-on.* An agent that can pick things up can name *the-object-I-am-holding.* The objects that are designated by these expressions can be directly "gotten at" by perception. It should be easy to answer the question of whether *the-object-I-am-fixated-on* is red.

Given a few primitive deictic names, such as those suggested above, we can make compound deictic expressions using directly perceptible relations. So, for example, we might speak of

the-object-on-top-of(the-object-I-am-fixated-on)

or

the-color-of(the-object-to-the-left-of(

the-object-I-am-fixated-on))

We can evaluate predicates, such as

bigger-than(the-object-I-am-fixated-on,

the-object-to-the-left-of(the-object-I-am-fixated-on))

(Benson 1996) describes an interesting system for synthesizing complex deictic expressions on demand.

One interesting property of these expressions is that they are *perceptually effective*; that is, that the agent can always foveate (or mark) them by taking a sequence of perceptual actions that is derived directly from the expression. It is easy for the agent to test for equality of two of these expressions by: following the first expression, putting a marker on the resulting object, following the second expression, putting a marker on the resulting object, and then testing to see whether the two markers are on the same object. Not all deictic expressions are perceptually effective. We are all familiar with situations in which *the-box-I-put-my-keys-in* or *my-glasses* are not perceptually effective.

Another important property of these deictic representations is that they implicitly perform a great deal of generalization. It is true, for instance, that *holding(the-object-I-am-holding)* no matter what particular object I am holding. Using deictic representations, we have a method for naming, and for generalizing over, objects without resorting to *ad hoc* names like *block5.*

Deixis can also be used in the action space. Rather than have an action like *pickup(block)* that takes the particular block to be picked up as a parameter, we might instead have a single action, *pickup* that always picks up the block that is currently fixated (or that has the "action" marker on it). In such a situation, we might be able to learn the rule:

$$red(\textit{the-block-I-am-fixated-on}) \wedge pickup \rightarrow$$

$$next \;\; red(\textit{the-block-I-am-holding})$$

Although it has no quantifiers, it expresses a universal property of all blocks.[3]

[3]This example illustrates a number of other questions we'll have to deal with eventually. We happen to know that, in fact, the block I am holding now is *the same block* as the one that I was fixated on a minute ago. How we can learn that is not clear. It requires a strong inductive leap. Even if we can't learn that, we'd like to learn the second-order fact that all of the properties of the block I'm holding now held of the block I was fixated on a minute ago.

Partial Observability When the agent is unable to observe the entire true state of the world at each time step, it must remember something about its history of actions and observations in order to behave effectively in the world. There are two ways to think about this aggregation: histories and belief states.

In the history-based model, the agent really does remember all, or as much of its history as it can. It may be selective about what it remembers, but the knowledge is represented in its historical context: "two time steps ago I was looking at a red block," or "the last block I was holding was green."

An alternative method is to aggregate the perceptual information into a *belief state*, which encodes what the agent currently knows about the state of the world. The fact that two time steps ago I was looking at a red block may actually mean that there is a red block to the left of my hand. Aggregating information into belief states may perform a useful kind of generalization, in that two different observation sequences with the same information content would result in the same belief state.

These two styles of representation may be equivalent at an information-theoretic level; however, the choice between them seems to have important consequences for learning and planning. An important part of this research program will be to study further the relative advantages and disadvantages of these two approaches.

The Role of Probability An independent representational question is whether, and, if so, how, to represent the uncertainty in the domain. There may be uncertainty in the agent's perception, in the effects of its actions, or in the dynamics of the environment, independent of the agent's actions.

Markov models capture uncertain world dynamics with atomic state representations. Dynamic Bayesian networks do the same for propositional representations. There is some work on extending Bayesian networks to probabilistic relational models (Koller & Pfeffer 1997; Getoor *et al.* 1999), but these models are not yet rich enough to capture probabilistic world dynamics we're interested in.

These models allow representation of any probabilistic world dynamics; to the extent that there are independence relations in the propositional and relational cases, the models can be represented compactly. However, there is considerable overhead in learning and working with such models. On the other hand, strictly logical models have serious problems with inconsistency when applied to real environments.

Real world environments tend to be neither completely deterministic nor so unpredictable that actions have a huge number of possible outcomes. Of course, clever choice of the terms used in articulating the world dynamics can mean the difference between a nearly deterministic and a highly stochastic world model: consider the difference between describing the location of a book as in my backback versus as occupying some particular volume in three-dimensional space. Similarly, the result of putting a block on top of a tower may be that the block stays there, or that the tower falls over. At one level of description, the tower falling over is an enormous number of possible arrangements of blocks, each with its own probability. It is neither possible, nor even desirable, to learn the world model at this level of abstraction.

We would like to find a representation that is appropriate for this middle ground. One idea would be to learn and use a model that is a deterministic idealization of the true stochastic world model, and to degrade gracefully when the idealization is violated. Rather than modeling all possible outcomes of an action, we might simply model a few of the most likely ones (which will probably cover most of the probability mass). And when a non-modeled outcome occurs, the agent will simply have to react to the new circumstances.

Learning Algorithms

In this section we consider which learning algorithms would be appropriate for learning policies (possibly via a value function) and world models, based on the choice of representation of observations.

Policy learning Most current research in reinforcement learning addresses the problem of learning in a propositional representation. The goal is to learn a value function, mapping state of the world into long-term values, but to do so in a compact way that affords generalization. Two main approaches are *neurodynamic programming* (NDP) (Bertsekas & Tsitsiklis 1996) and decision-tree methods (Chapman & Kaelbling 1991). In NDP, the value function is stored in a feed-forward neural network. An alternative is to incrementally build a decision tree, with values stored at the leaves. An advantage of this method is that it has a direct extension to the partially observable case; in addition, it makes clear which of the propositional attributes are important.

There has been very little work on relational reinforcement learning. One promising idea is to adapt methods from inductive logic programming to the task (Dzeroski, de Raedt, & Blockeel 1998), but the initial work still exhibits fairly weak generalization abilities.

The question of most interest to us, for now, is how to do reinforcement learning with a deictic representation, which is by its very nature partially observable. There are two general strategies for learning in partially observable environments. The first is to learn policies directly, typically by performing local search in the space of policies (pursued by (Baum & Durdanovic 2000) in a similar domain). The second is to try to reconstruct the underlying world state from histories of actions and observations, then to do standard propositional RL on a representation of the reconstructed state. We feel that this second approach will be more generally applicable, and offers more opportunity for between-task transfer. We are initially using McCallum's (McCallum 1995) UTREE algorithm, which is an extension of decision-tree reinforcement-learning that also considers properties of historic actions and observations.

World model Learning a world model is fundamentally a supervised learning problem, making it easier, at some level, than the policy-learning problem. However, as we mentioned earlier, the entire world dynamics model will be huge in any interesting domain. It will be important to use the agent's goals to focus the model learning so that only the useful and important aspects of the world dynamics are represented.

There has been a lot of work on supervised learning in the propositional case. Methods that focus on learning prediction rules for dynamic events include the method of (Oates, Jensen, & Cohen 1998) for generating association rules to predict asynchronous events, the schema mechanism of (Drescher 1991), and Bayesian network learning methods (Heckerman 1995) applied to dynamic Bayesian networks (Dean & Kanazawa 1989). Each of these methods has strengths and weaknesses. The schema method and association-rule learning are appealing because they learn rules independently, allowing the agent to accrete knowledge of the model over time, initially ignoring many aspects. However, the probabilistic foundations of learning and reasoning in this way are much weaker (it is not clear exactly how the rules specify a joint probability distribution over outcomes, for example). The Bayesian network method is much stronger foundationally, but does not have built-in frame assumptions and is harder to use incrementally.

These methods can all deal with partial observability to some degree, but not sufficiently well for our purposes. Thus, an important part of our research is to find a method of incremental model learning in partially observable propositional domains, and to characterize formally its probabilistic semantics.

Ultimately, of course, we would like to extend this to the relational and to the restricted quantified case. To do so, we will build on work from probabilistic ILP (Muggleton forthcoming) and relational Bayesian networks (Getoor *et al.* 1999).

References

Baum, E. B., and Durdanovic, I. 2000. Evolution of cooperative problem-solving in an artificial economy. `http://www.nec.com/homepages/eric/hayek32000.ps`.

Benson, S. 1996. *Learning Action Models for Reactive Autonomous Agents.* Ph.D. Dissertation, Stanford University.

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming.* Belmont, Massachusetts: Athena Scientific.

Chapman, D., and Kaelbling, L. P. 1991. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

Chapman, D. 1991. *Vision, Instruction, and Action.* Cambridge, Massachusetts: The MIT Press.

Dean, T., and Kanazawa, K. 1989. A model for reasoning about persistence and causation. *Computational Intelligence* 5:142–150.

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

Dzeroski, S.; de Raedt, L.; and Blockeel, H. 1998. Relational reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 136–143. Morgan Kaufmann.

Getoor, L.; Friedman, N.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence.*

Halpern, J. Y. 1990. An analysis of first-order logics of probability. *Artificial Intelligence* 46:311–350.

Heckerman, D. 1995. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, Washington. Revised June 96. Available at `ftp://ftp.research.microsoft.com/pub/techreports/winter94-95/tr-95-06.ps`.

Koller, D., and Pfeffer, A. 1997. Object oriented bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertain ty in Artificial Intelligence.*

McCallum, A. K. 1995. *Reinforcement Learning with Selective Perception and Hidden State.* Ph.D. Dissertation, University of Rochester, Rochester, New York.

Muggleton, S. H. forthcoming. Statistical aspects of logic-based machine learning. *ACM Transactions on Computational Logic.*

Nilsson, N. J. 1986. Probabilistic logic. *Artificial Intelligence* 28:71=87.

Oates, T.; Jensen, D.; and Cohen, P. R. 1998. Discovering rules for clustering and predicting asynchronous events. In Danyluk, A., ed., *Predicting the Future: AI Approaches to Time-Series Problems.* AAAI Press. 73–79. Technical Report WS-98-07.

Smith, B. C. 1996. *On the Origin of Objects.* Cambridge, Massachusetts: The MIT Press.

Triesman, A. 1985. Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing* 31:156–177.

Ullman, S. 1984. Visual routines. *Cognition* 18:97–159.

Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. *Machine Learning* 8(3):279–292.

Whitehead, S. D., and Ballard, D. H. 1991. Learning to perceive and act by trial and error. *Machine Learning* 7(1):45–83.