Learning Distinctions and Rules in a Continuous World through Active Exploration

Jonathan Mugan
Computer Science Department
University of Texas at Austin
Austin Texas, 78712 USA
jmugan@cs.utexas.edu

Benjamin Kuipers
Computer Science Department
University of Texas at Austin
Austin Texas, 78712 USA
kuipers@cs.utexas.edu

Abstract

We present a method that allows an agent through active exploration to autonomously build a useful representation of its environment. The agent builds the representation by iteratively learning distinctions and predictive rules using those distinctions. We build on earlier work in which we showed that by motor babbling an agent could learn a representation and predictive rules that by inspection appeared reasonable. In this paper we add active learning and show that the agent can build a representation that allows it to learn predictive rules to reliably control its hand and to achieve a simple goal.

1. Introduction

The challenge is to build a robot that can learn about itself and its environment in the same way that children do. In Piaget's [1952] theory of cognitive development, children constructed this knowledge in stages. More recently, Cohen [2002] has proposed an information processing approach to cognitive development in which children are endowed with a domain-general information processing system that they use to bootstrap knowledge.

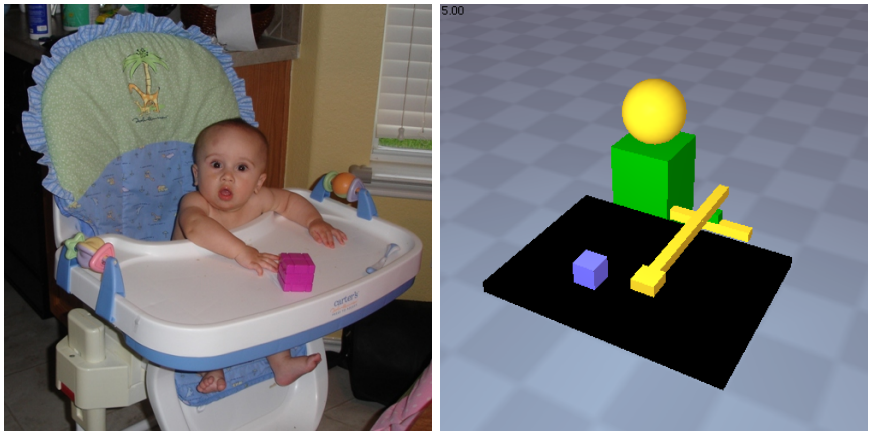
In this work we focus on how a developing agent can learn temporal contingencies in the form of predictive rules over events. There is evidence that infants are able to detect contingencies shortly after birth [DeCasper and Carstens, 1981]. Watson [2001] proposed a model of contingencies based on his observations of infant behavior. In this model, a *prospective* temporal contingency is one in which an event B tends to follow an event A with a likelihood greater than chance, and a *retrospective* temporal contingency is one in which an event A tends to come before an event B more often than chance.

Watson recognized that an impediment to learning contingencies is finding the distinctions necessary to determine when an event has occurred. Watson's proposes looking for new distinctions when the probability associated with a prospective contingency on two events does not match the probability associated with the retrospective contingency on those same events. He also uses this mismatch in probabilities to indicate that a contingency may only hold in a certain situation.

Drescher [1991] proposed a model of contingencies inspired by Piaget, he refers to contingencies as *schemas* and he finds these schemas by a process called *marginal attribution*. Marginal attribution first finds results that follow actions in a method similar to Watson's prospective probabilities. Then for each schema in the form of an action and a result the algorithm searches for a *context* (situation) that makes the result more likely to follow that action.

We represent both prospective contingencies and contingencies in which two events happen simultaneously using predictive rules. These predictive rules are learned using a method inspired by marginal attribution, but we move beyond Drescher by working with continuous variables. This brings up the issue, pointed out by Watson, of determining when an event has occurred. For each predictive rule we look for a new distinction that would make it more reliable. And although we do not explicitly represent retrospective contingencies, for each event we look for a new distinction that would allow us to predict that event. These new distinctions allow the agent both to learn more accurate contingencies and to perceive new events that allow it to learn new contingencies.

In [Mugan and Kuipers, 2007] we developed an algorithm in which the agent used motor babbling to learn distinctions and contingencies. However, using undirected motor babbling does not allow learning to scale to larger problems because too much effort is wasted on uninteresting portions of the state space. In this paper the algorithm is expanded to allow the agent to purposefully explore its environment and to learn as it seeks to achieve goals. This allows us to demonstrate that the learned representation is useful



(a) 6-month-old baby (b) simulated robot

Figure 1: We use the situation of six-month-old baby sitting in a highchair (a) as the setup to evaluate our method using a simple "baby robot" (b). The robot is implemented in Breve [Klein, 2003]. It has a torso with a 2-dof arm and is sitting in front of a tray with a block.

by measuring the agent's increasing ability to achieve a simple goal.

In the implementation of this algorithm, the agent receives as input the values of time-varying continuous variables, but the agent can only represent, reason about, and construct knowledge using discrete values. The agent discretizes its input using distinctions in the form of *landmarks*. Landmarks are used to create a discrete variable $v(t)$ for each continuous variable $\tilde{v}(t)$. If the real value of $\tilde{v}(t)$ falls between two landmarks v_1 and v_2, then the discrete variable $v(t)$ will have the open interval between v_1 and v_2 as its value, and $v(t) = (v_1, v_2)$. Once $v(t)$ is associated with $\tilde{v}(t)$, the agent can then focus its attention on changes in the discrete value of v, called *events*.

The agent greedily learns rules that use one event to predict another. By storing the real values of the variables used in a predictive rule each time the rule is observed to be applicable in the environment, the agent can use the success or failure of the rule as a supervisory signal to learn more landmarks through standard discretization techniques such as [Fayyad and Irani, 1993]. These new distinctions in the form of landmarks allow the agent to learn more predictive rules, which in turn can lead to more landmarks. Predictive rules can be made more deterministic by adding context variables. The context of a rule indicates when the rule will make accurate predictions and when it will not.

We evaluate our algorithm using the simulated robot shown in Fig. 1(b). The setup for the robot is taken from the situation of a baby sitting at a high chair, shown in Fig. 1(a). The value of this work is that it provides a method for an autonomous agent to break up the world at its joints. The agent learns the distinctions that are relevant to the way that it interacts with the world. In this paper we evaluate these distinctions by how well they allow the agent to reach out and move a block.

2. Knowledge Representation and Learning

As described in [Mugan and Kuipers, 2007], a critical task for the learning agent is to learn appropriate abstractions from continuous to discrete variables. Initially, the values of the continuous variables are completely meaningless. Our goal is for the agent to learn, from its own experience, to identify *landmark values* that make important qualitative distinctions for each variable. The importance of a qualitative distinction is estimated from the reliability of the rules that can be learned, given that distinction.

The qualitative representation is based on QSIM [Kuipers, 1994]. For each continuous variable $\tilde{x}(t)$ two discrete variables are created: a discrete variable $x(t)$ that represents the magnitude of $\tilde{x}(t)$, and a discrete variable $\dot{x}(t)$ that represents the direction of change of $\tilde{x}(t)$. (Non-zero directions of change that persist fewer than three time-steps are filtered out.)

A continuous variable $\tilde{x}(t)$ ranges over some subset of the real number line $(-\infty, +\infty)$. In QSIM, its magnitude is abstracted to a discrete variable $x(t)$ that ranges over a *quantity space* $Q(x)$ of qualitative values. $Q(x) = L(x) \cup I(x)$, where $L(x) = \{x_1, \cdots x_n\}$ is a totally ordered set of landmark values, and $I(x) = \{(-\infty, x_1), (x_1, x_2), \cdots (x_n, +\infty)\}$ is the set of mutually disjoint open intervals that $L(x)$ defines in the real number line. A quantity space with two landmarks might be described by (x_1, x_2), which implies five distinct qualitative values, $Q(x) = \{(-\infty, x_1), x_1, (x_1, x_2), x_2, (x_2, +\infty)\}$.

A discrete variable $\dot{x}(t)$ representing the direction of change of $\tilde{x}(t)$ has a single intrinsic land-

mark at 0, so its initial quantity space is $Q(\dot{x}) = \{(-\infty, 0), 0, (0, +\infty)\}$. Initially, when the agent knows of no meaningful qualitative distinctions among values for $\tilde{x}(t)$, we describe the quantity space as the empty list of landmarks, (). (Note that because we evaluate the algorithm with a discrete-timestep simulator, if x_1 is a landmark and $\tilde{x}(t-1) < x_1$ and $\tilde{x}(t) > x_1$ then $x(t) = x_1$.) Table 1 lists the variables the "baby robot" knows about, and their initial and final landmarks, the meaning of these variables is explained in Section 3. Note that for most magnitude variables, zero is just another point on the number line, so those variables initially have no landmarks.

2.1 Events

If a is a qualitative value of a discrete variable A, meaning $a \in Q(A)$, then the *event* $A_t \rightarrow a$ is defined by $A(t-1) \neq a$ and $A(t) = a$. That is, an event takes place when a discrete variable A changes to value a at time t, from some other value. We will often drop the t and describe this simply as $A \rightarrow a$. We will also refer to an event as E when the variable and qualitative value involved are not important.

Our goal is for the agent to learn predictive rules and landmarks to describe regularities in the occurrence of events.

2.2 Predictive Rules

Temporal contingencies are described using predictive rules. Consider a subset of the scenario shown in Fig. 1(b). The continuous variable $\tilde{h}_x$ gives the location of the hand in the x direction, and the continuous variable $\tilde{u}_x$ gives the motor force applied in the x direction. We would like the agent to learn a rule that predicts $\dot{h}_x \rightarrow (0, \infty)$, the event that the hand begins to move to the right. The agent will look at all other events and find that if a force is given in the positive x direction, event $u_x \rightarrow (0, \infty)$, then the event $\dot{h}_x \rightarrow (0, \infty)$ is more likely to occur than it would otherwise. It will then create a rule of the form $r_1 = \langle u_x \rightarrow (0, \infty) \Rightarrow \dot{h}_x \rightarrow (0, \infty) \rangle$.

In the simulator it takes a force of 300 to move the hand. By noting the real value $\tilde{u}_x$ each time event $u_x \rightarrow (0, \infty)$ occurs, the agent can use the occurrence or nonoccurrence of event $\dot{h}_x \rightarrow (0, \infty)$ as a supervisory signal and create a new landmark at $\tilde{u}_x = 300$. It can then update r_1 to be $r_1 = \langle u_x \rightarrow (300, \infty) \Rightarrow \dot{h}_x \rightarrow (0, \infty) \rangle$.

The rule r_1 is still not completely deterministic because if the hand is already all the way to the right then it cannot go any farther, even if event $u_x \rightarrow (300, \infty)$ occurs. But initially, the agent has no way of reasoning about the location of the hand because $\tilde{h}_x$ has no landmarks. In the simulator, the maximum value for $\tilde{h}_x = 2.0$, so by storing the value $\tilde{h}_x$ in r_1 each time event $u_x \rightarrow (300, \infty)$ occurs, the agent can find that as long as $\tilde{h}_x \neq 2.0$ event $\dot{h}_x \rightarrow (0, \infty)$ almost always occurs. It then learns a landmark on $\tilde{h}_x = 2.0$ which allows it to add h_x as a context to r_1 giving $r_1 = \langle \{h_x\} : u_x \rightarrow (300, \infty) \Rightarrow \dot{h}_x \rightarrow (0, \infty) \rangle$. Putting h_x in the context of r_1 allows the agent to determine if event $\dot{h}_x \rightarrow (0, \infty)$ will follow $u_x \rightarrow (300, \infty)$ by looking at the value of $h_x(t)$.

Rule r_1 is an example of a causal predictive rule. There are two types of predictive rules: *causal rules* represent that one event occurs after another later in time, the linking of events appearing as causal to the agent; and *functional rules* represent that two events are linked by a function and so happen at the same time. For both types of rules we focus only on those that predict positive or negative changes in direction of change variables. These two types of predictive rules differ only in the time component, so after initially discussing them separately, we will simply refer to both types as predictive rules.

2.3 Causal Rules

We now formally describe causal rules. A *causal rule* r has the form $\langle \mathcal{C} : E_1 \Rightarrow E_2 \rangle$, where $E_1(t)$ is one event, say $A_t \rightarrow a$, $E_2(t')$ is another event over a direction of change variable, say $B_{t'} \rightarrow b$, that takes place relatively soon after t, and the *context* $\mathcal{C}$ is a set of discrete magnitude variables. That E_2 takes place "relatively soon after" $E_1(t)$ is formalized in terms of an integer time-delay $k = 6$.

$$soon(t, E_2) \quad \equiv \quad \exists t' \; [t < t' < t + k \; \wedge \; E_2(t')] \quad (1)$$

A rule $r = \langle \mathcal{C} : E_1 \Rightarrow E_2 \rangle$ is *activated* when $E_1(t)$ occurs and *succeeds* when:

$$succeeds(r, t) \quad \equiv \quad E_1(t) \; \wedge \; soon(t, E_2) \quad (2)$$

Associated with a causal rule $r = \langle \mathcal{C} : E_1 \Rightarrow E_2 \rangle$ is a probability distribution of the form

$$P(soon(t, E_2) | E_1(t) = \mathsf{true}, \mathcal{C}(t-1)) \quad (3)$$

which is the conditional probability distribution over the binary random variable $soon(t, E_2)$, given that $E_1(t)$ is true and the values of the variables in $\mathcal{C}$ at time $t - 1$.

The agent greedily searches for rules that deterministically predict when one event will follow another. We use the *entropy* of a rule as a measure of how deterministic it is. We define the entropy of a rule $r = \langle \mathcal{C} : E_1 \Rightarrow E_2 \rangle$ as the conditional entropy of $soon(t, E_2)$ given $\mathcal{C}(t)$, with the added restriction that event $E_1(t)$ occurs. In equation form it is

$$H(r) = H(soon(t, E_2) | E_1(t) = \mathsf{true}, \mathcal{C}(t-1)). \quad (4)$$

However, a rule r can have low entropy if it predicts that E_2 will almost never follow E_1, so a rule must

have more than low entropy to be useful. We define a concept called *best reliability* represented as $brel(r)$. For rule r, $brel(r)$ is the highest probability of success for any value of $\mathcal{C}$. If $\mathcal{C} = \emptyset$ then $brel(r)$ is just the probability of success of r.

2.3.1 Learning a Causal Rule

The agent starts by searching for two events E_1 and E_2 such that observing event E_1 means that event E_2 is significantly more likely to occur than it would have been otherwise.

The agent asserts an initial rule $\langle \emptyset : E_1 \Rightarrow E_2 \rangle$ with empty context, when $Pr(soon(t, E_2)|E_1(t)) > 0.1$ and

$$\iota(Pr(soon(t, E_2)|E_1(t)), Pr(soon(t, E_2))) > \theta_a \quad (5)$$

where the function on probabilities $\iota(p, q) = \frac{p}{q} \cdot \frac{1-q}{1-p}$ has been defined to have higher resolution near the extremes, and lower resolution near the center, over the interval $(0, 1)$ of probability values. The parameter θ_a specifies how much more likely E_2 should be, after E_1 has been observed. (Here and elsewhere we require a minimum number of relevant observations so the probability will be reliable.)

2.3.2 Learning a Context for a Causal Rule

Once the agent has learned a rule $r = \langle \emptyset : E_1 \Rightarrow E_2 \rangle$, it searches for a discrete magnitude variable v_1 such that if r is modified to be $r' = \langle \{v_1\} : E_1 \Rightarrow E_2 \rangle$ the variable v_1 provides sufficient information gain

$$H(r) - H(r') \quad > \quad \theta_{ig}. \quad (6)$$

The parameter θ_{ig} determines how much information gain is required to augment the context. If there are multiple discrete variables that meet this criterion, then the one providing the largest information gain is chosen.

Using an approach inspired by Drescher [1991], once the agent has learned a rule $r' = \langle \{v_1\} : E_1 \Rightarrow E_2 \rangle$ it searches for another discrete magnitude variable v_2 such that if r' is modified to be $r'' = \langle \{v_1, v_2\} : E_1 \Rightarrow E_2 \rangle$ the variable v_2 provides sufficient information gain $H(r') - H(r'') > \theta_{ig}$. In principle, an arbitrarily large context can be learned, but in this implementation the size is limited to two.

2.4 Functional Rules

A *functional rule* $r = \langle \mathcal{C} : E_1 \Rightarrow E_2 \rangle$ has the same form as a causal rule, and behaves in a similar way, with three exceptions. The first difference is in the timing of the events: the predicate $soon(t, E_2)$ is replaced with $E_2(t)$, which means that the events E_1 and E_2 must happen in the same timestep. The second difference is that functional rules are not used to learn landmarks. And the third difference is because there is no time delay. If a functional rule $\langle \mathcal{C} : E_1 \Rightarrow E_2 \rangle$ is learned but its opposite $\langle \mathcal{C} : E_2 \Rightarrow E_1 \rangle$ has a significantly higher rate of success before the context is considered, then $\langle \mathcal{C} : E_1 \Rightarrow E_2 \rangle$ is replaced by $\langle \mathcal{C} : E_2 \Rightarrow E_1 \rangle$. We will refer to both types of rules simply as *rules*.

2.5 Learning a New Landmark

Inserting a new landmark x^* into (x_i, x_{i+1}) allows that interval to be replaced in $Q(x)$ by two intervals and the dividing landmark: (x_i, x^*), x^*, (x^*, x_{i+1}). Adding this new landmark into the quantity space $Q(x)$ allows a new distinction to be made that may transform r into a new rule r'. (When a new landmark x^* is learned we throw out the statistics for (x_i, x_{i+1}) and start fresh with (x_i, x^*), x^*, (x^*, x_{i+1}), however this means that we must also check that the reliability of r does not significantly deteriorate with an improvement in $H(r)$.) A new landmark can be learned either by improving a predictive rule or by reliably preceding an event leading to a new predictive rule.

2.5.1 Landmarks that Improve Rules

If a landmark candidate for a rule $r = \langle \mathcal{C} : A \rightarrow b \Rightarrow B \rightarrow b \rangle$ is on variable A, then the landmark must improve the best reliability of r to be adopted. If the landmark is on another variable then it must improve the entropy of r to be adopted by modifying $\mathcal{C}$.

For an example of a landmark that improves the best reliability, recall the initial incarnation of our rule $r_1 = \langle u_x \rightarrow (0, \infty) \Rightarrow \dot{h}_x \rightarrow (0, \infty) \rangle$. The algorithm first learned a landmark on $\tilde{u}_x$ at 300. With this landmark rule r_1 could be modified to be $r_1' = \langle u_x \rightarrow (300, \infty) \Rightarrow \dot{h}_x \rightarrow (0, \infty) \rangle$, which had a higher rate of success. In general, we add a new landmark to the quantity space $Q(A)$ when it increases the best reliability of r by transforming it into r' so that $\iota(brel(r'), brel(r)) > \theta_a$.

For an example of a landmark that improves the entropy of rule, recall that rule $r_1 = \langle u_x \rightarrow (300, \infty) \Rightarrow \dot{h}_x \rightarrow (0, \infty) \rangle$ was further improved by learning a landmark on $\dot{h}_x$ at 2.0. This allowed the agent to make a distinction on h_x and to add it to the context of r_1 improving its entropy. In general, a new landmark is added to the quantity space $Q(x)$ when it makes a rule r more deterministic by transforming it into r' so that $H(r) - H(r') > \theta_{ig}$.

Landmark candidates are chosen considering the number of data points in the interval and the highest gain [Fayyad and Irani, 1993]. Depending on the relative gains of nearby potential values for a new landmark x^*, this search can result in either a precise numerical value, or a range of possible values for x^* on different occasions: $range(x^*) = [lb, ub]$.

Examples of both cases are shown in Table 1.

If a landmark candidate improving a rule is adopted, then its location is continually updated as the agent learns more.

2.5.2 Landmarks at Events

For each event E a histogram is maintained for each continuous variable $\tilde{v}$. Each time E occurs the histogram is updated with the current value of $\tilde{v}$. A landmark candidate is created for $\tilde{v}$ when the distribution of $\tilde{v}$ when E occurs is significantly different from its background distribution. The location of the landmark is taken to be the middle of the histogram bucket where the difference between distributions is the greatest. This landmark candidate is adopted as a new landmark if it leads to a rule that predicts the event E.

2.6 Active Learning

Active learning allows the agent to systematically explore parts of the state space that it might too infrequently explore using random movements. In this work the agent engages in active learning by continually selecting a goal and then working to achieve it. A goal g is of the form $g = B{\rightarrow}b$ where B is a discrete variable and $b \in Q(B)$ is a qualitative value. Goal g is *achieved* at timestep t if $B(t) = b$.

Actively exploring the world requires forming a plan. We use a simple recursive planning algorithm similar to STRIPS [Nilsson, 1980] shown in Fig. 2. The agent uses the SelectRule function to find a rule $r = \langle \mathcal{C} : A{\rightarrow}a \Rightarrow B{\rightarrow}b \rangle$ where $B{\rightarrow}b$ is the event on the top of the stack. If no such rule is found but top-of-stack is an event on a magnitude variable x, then a new goal is pushed onto the stack based on the direction of change variable $\dot{x}$.

If a rule r is found then the agent must determine if its context $\mathcal{C}$ is satisfied using the Satisfied function. In this implementation a context $\mathcal{C}$ consists of either a single discrete variable or a pair of discrete variables and gives the probability of success of r for each value of its variables at activation. The Satisfied function returns true if the probability of success of r given by $\mathcal{C}$ for the current state of the world is greater than 0.6.

If the context $\mathcal{C}$ is not satisfied, a subgoal for $\mathcal{C}$ is chosen using the SelectEvent function. If $\mathcal{C}$ consists of one variable v then it returns $v{\rightarrow}q$ where $q \in Q(v)$ maximizes the probability of the success of r given the current state of the world. If $\mathcal{C}$ consists of two variables v_1 and v_2, then the function checks to see if the context can be satisfied by setting one of v_1 or v_2 to a value q. If this is so, say for v_1, then SelectEvent returns $v_1{\rightarrow}q$. If the context cannot be satisfied by setting only one of the variables then it finds the context value $\langle v_1 = q_1, v_2 = q_2 \rangle$ with the maximum probability of success for r and by random choice one of $v_1{\rightarrow}q_1$ or $v_2{\rightarrow}q_2$ is returned.

```
input: goal
push(goal)
loop
  r := SelectRule(⟨C : A→a ⇒ top-of-stack⟩)
  if none then
    let x→q := top-of-stack
    if x is a magnitude variable then
      if A(t) < a then
        push(ẋ→(0, ∞))
        continue
      else if A(t) > a then
        push(ẋ→(−∞, 0))
        continue
      else
        FAIL
      end if
    else
      FAIL
    end if
  end if
  if Satisfied(C) then
    if A→a ≡ U→u then
      PerformAction(U→u,stack)
      return
    else
      push(A→a)
    end if
  else
    push(A→a)
    push(SelectEvent(C))
  end if
end loop
```

Figure 2: A basic planning algorithm. Details for the functions SelectRule, PerformAction, Satisfied, and SelectEvent are given in Section 2.6

If at any time the planner attempts to push a goal g that is already on the stack, or if g is on a direction of change variable and its opposite is on the stack, then the planner returns FAIL.

When a motor variable is encountered as the left hand side of r, the agent uses the PerformAction function to carry out the action. For the motor command $U{\rightarrow}u$ a force amount $\tilde{u}$ is chosen randomly using a uniform distribution over finite $range(u)$, and all other motor variables are set to a force of 0 (an event that is ignored by the rule learner). PerformAction is in the form of a test-operate-test-exit (TOTE) unit [Miller et al., 1960] and the motor value is maintained until the exit criterion is satisfied. To determine how long this motor value should be maintained the agent looks at the stack. The agent watches the highest subgoal in the stack that is associated with a

magnitude variable and if that subgoal is achieved it terminates the action immediately if the subgoal was from the context of a rule and in k timesteps otherwise. (If no such subgoal on a magnitude variable exits it watches the highest subgoal on a direction of change variable and terminates the action after it is achieved for three consecutive timesteps.) The agent also watches the highest subgoal in the stack that is associated with a direction of change variable. If that subgoal is initially achieved and then later is no longer achieved the agent assumes the plan is off track and termites the action. The action is also terminated if the real value of at least one variable does not change at each timestep, or if 40 timesteps pass.

2.6.1 Goal Selection

Initially, the agent explores by motor babbling, but as the agent learns more the probability of it choosing a goal and purposefully pursuing it increases. A new goal is chosen from a set of candidate goals, the set of candidate goals is determined by a set M of discrete variables. For each discrete variable $v \in M$ a candidate goal $g = v{\rightarrow}q$ is created for each $q \in Q(v)$.

When a goal $g = v{\rightarrow}q$ is chosen it is sent to the planning module, the goal is achieved if $v = q$ when the planning module returns. The agent chooses the candidate goals in succession until each goal is chosen $m = 20$ times. After each goal has been chosen m times, goals are chosen based on a learnability score, with the goal with the highest score chosen at each opportunity. The learnability score of a goal g is determined by creating a vector of the success or failure of the last m activations of g, and then taking the entropy of that vector.

The learnability score is inspired by [Schmidhuber, 1991], a low entropy indicates that the agent can either consistently achieve g and does not stand to learn much by trying more, or the agent is not having much success achieving g and therefore it is currently too difficult for the agent to learn anything new by trying to achieve it.

2.7 The Learning Process

During the learning process the algorithm builds a stratified model on the discrete variables. Each stratum $\mathcal{S}_i$ consists of a set of discrete variables. The purpose of the stratified model is serve as a focus of attention and to constrain the proliferation of rules by favoring those that emanate from the agent as the causal source of events.

Each discrete variable can reside in at most one stratum. Rules can only be learned that use an event on a variable in stratum $\mathcal{S}_i$ to predict an event on a variable in stratum $\mathcal{S}_i$ or $\mathcal{S}_{i+1}$ (with the exception of a functional rule $\langle \mathcal{C} : E_2 \Rightarrow E_1 \rangle$ being learned because it has a higher rate of success than $\langle \mathcal{C} : E_1 \Rightarrow E_2 \rangle$).

Table 1: Variables, their ranges of values, and initial and final landmarks

Var.	Range	Initial	Final	Landmarks
u_x	$[-500, 500]$	(0)	$(L_1, 0, L_2)$	$L_1 = -301.51$
u_y	$[-500, 500]$	(0)	$(L_3, 0, L_4)$	$L_2 = 297.71$
h_x	$[-2.0, 2.0]$	()	(L_5, L_6)	$L_3 = -298.54$
$\dot{h}_x$	$(-\infty, +\infty)$	(0)	(0)	$L_4 = 300.18$
h_y	$[-2.0, 2.0]$	()	(L_7, L_8, L_9)	$L_5 = [-2.04, -1.96]$
$\dot{h}_y$	$(-\infty, +\infty)$	(0)	(0)	$L_6 = [1.93, 2.04]$
h_a	$\{0, 1\}$	Binary	Binary	$L_7 = [-2.01, -1.97]$
b_x	$(-\infty, +\infty)$	()	(L_{10}, L_{11}, L_{12})	$L_8 = -0.54$
$\dot{b}_x$	$(-\infty, +\infty)$	(0)	(0)	$L_9 = [1.98, 2.01]$
b_y	$(-\infty, +\infty)$	()	(L_{13}, L_{14})	$L_{10} = -8.61$
$\dot{b}_y$	$(-\infty, +\infty)$	(0)	(0)	$L_{11} = -0.14$
b_a	$\{0, 1\}$	Binary	Binary	$L_{12} = 3.02$
c_x	$(-\infty, +\infty)$	()	(L_{15}, L_{16}, L_{17})	$L_{13} = 2.75$
$\dot{c}_x$	$(-\infty, +\infty)$	(0)	(0)	$L_{14} = 2.83$
c_y	$(-\infty, +\infty)$	()	(L_{18}, L_{19}, L_{20})	$L_{15} = -2.08$
$\dot{c}_y$	$(-\infty, +\infty)$	(0)	(0)	$L_{16} = 2.07$
e	$[0, +\infty)$	()	(L_{21}, L_{22})	$L_{17} = 6.64$
$\dot{e}$	$(-\infty, +\infty)$	(0)	(0)	$L_{18} = -2.04$
				$L_{19} = -0.97$
				$L_{20} = -0.52$
				$L_{21} = 0.10$
				$L_{22} = 0.23$

Table 2: Learned Rules (T indicates either a causal rule or functional rule)

Strata	T	Rule
$\mathcal{S}_0 =$ $\{u_x,\ u_y\}$	C	$\{h_x\} : u_x{\rightarrow}(-\infty, -303.31) \Rightarrow \dot{h}_x{\rightarrow}(-\infty, 0)$
	C	$\{h_x\} : u_x{\rightarrow}(300.78, +\infty) \Rightarrow \dot{h}_x{\rightarrow}(0, +\infty)$
	C	$\{b_x\} : u_y{\rightarrow}(301.64, +\infty) \Rightarrow \dot{h}_y{\rightarrow}(0, +\infty)$
	C	$\{h_y\} : u_y{\rightarrow}(-\infty, -293.54) \Rightarrow \dot{h}_y{\rightarrow}(-\infty, 0)$
$\mathcal{S}_1 =$ $\{h_x,\ \dot{h}_x,$ $h_y,\ \dot{h}_y\}$	F	$\emptyset : h_x{\rightarrow}(-\infty, 0) \Rightarrow \dot{c}_x{\rightarrow}(-\infty, 0)$
	F	$\emptyset : \dot{h}_x{\rightarrow}(0, +\infty) \Rightarrow \dot{c}_x{\rightarrow}(0, +\infty)$
	F	$\emptyset : \dot{h}_y{\rightarrow}(-\infty, 0) \Rightarrow \dot{c}_y{\rightarrow}(-\infty, 0)$
	F	$\emptyset : \dot{h}_y{\rightarrow}(0, +\infty) \Rightarrow \dot{c}_y{\rightarrow}(0, +\infty)$
$\mathcal{S}_3 =$ $\{e,\ \dot{e},$ $c_x,\ \dot{c}_x,$ $c_y,\ \dot{c}_y\}$	C	$\{h_x\} : c_x{\rightarrow}[6.64] \Rightarrow b_a{\rightarrow}\mathsf{false}$
	F	$\{c_x, c_y\} : e{\rightarrow}[0.23] \Rightarrow \dot{b}_x{\rightarrow}(0, +\infty)$
	F	$\{c_x\} : \dot{c}_x{\rightarrow}(0, +\infty) \Rightarrow \dot{e}{\rightarrow}(0, +\infty)$
	C	$\{b_y, c_x\} : e{\rightarrow}[0.10] \Rightarrow \dot{b}_x{\rightarrow}(-\infty, 0)$
$\mathcal{S}_4 =$ $\{b_x,\ \dot{b}_x,$ $b_y,\ \dot{b}_y\}$	F	$\{h_y, c_x\} : \dot{b}_y{\rightarrow}(0, +\infty) \Rightarrow \dot{c}_y{\rightarrow}(-\infty, 0)$
	F	$\{b_y, c_x\} : \dot{b}_y{\rightarrow}(-\infty, 0) \Rightarrow \dot{b}_x{\rightarrow}(-\infty, 0)$
	F	$\{b_y, c_x\} : \dot{b}_y{\rightarrow}(0, +\infty) \Rightarrow \dot{b}_x{\rightarrow}(-\infty, 0)$

A discrete variable B is added to stratum $\mathcal{S}_{i+1}$ if B is currently not a member of any stratum and there is a rule r of the form $r = \langle \mathcal{C} : A{\rightarrow}a \Rightarrow B{\rightarrow}b \rangle$ where $A \in \mathcal{S}_i$. A magnitude variable and its derivative are always in the same stratum.

The initial stratum $\mathcal{S}_0$ consists of the motor primitives. From there, the agent builds its model by looping over the following sequence of steps

1. Do 7 times
   (a) Actively explore the world with the set of candidate goals coming from the discrete variables in $M = \mathcal{S}_i \cup \mathcal{S}_{i+1}$ for 1000 timesteps
   (b) Learn new causal and functional rules
   (c) Learn new landmarks by examining statistics stored in rules and events
2. Gather 3000 more timesteps of experience to solidify the learned rules
3. Update the strata
4. Goto 1

We call a rule *sufficiently deterministic* if it has both low entropy and high best reliability. To update the

strata the agent builds the next stratum and adjusts the earlier strata using the sufficiently deterministic rules. The agent also removes redundant rules stemming from a motor variable. Intuitively, if there are sufficiently deterministic rules $\langle E_1 \Rightarrow E_2 \rangle$, $\langle E_2 \Rightarrow E_3 \rangle$, and $\langle E_1 \Rightarrow E_3 \rangle$, then $\langle E_1 \Rightarrow E_3 \rangle$ is redundant and can be pruned. The agent then resumes the loop on the next stratum ($i = i + 1$). If this stratum $\mathcal{S}_i$ contains no variables the agent sets the stratum index i to the highest stratum that does contain variables.

3. Evaluation

3.1 Experimental Setup

The evaluation uses the simulation scenario shown in Fig. 1(b). The robot has two motor variables $\tilde{u}_x$ and $\tilde{u}_y$ that move the hand in the x and y direction respectively. The perceptual system creates variables for each of the two tracked objects in this environment: the hand and the block. The variables corresponding to the hand are $\tilde{h}_x(t)$, $\tilde{h}_y(t)$, and $h_a(t)$. The continuous variables $\tilde{h}_x(t)$ and $\tilde{h}_y(t)$ represent the location of the hand in the x and y directions, respectively, and the Boolean variable $h_a(t)$ represents whether the hand is in view. The variables corresponding to the block are $\tilde{b}_x(t)$, $\tilde{b}_y(t)$, and $b_a(t)$ and they have the same respective meanings as the variables for the hand. The relationship between the hand and the block is represented by the continuous variables $\tilde{c}_x(t)$, $\tilde{c}_y(t)$, and $\tilde{e}(t)$. The variables $\tilde{c}_x(t)$ and $\tilde{c}_y(t)$ represent the coordinates of the center of the hand in the frame of reference of the center of the block, and the variable $\tilde{e}(t)$ represents the distance between the hand and the block.

The conversion process from continuous to discrete produces the variables that the agent can represent and reason about. The motor variables are $u_x(t)$ and $u_y(t)$ controlling the hand. The state of the hand is given by $h_x(t)$, $h_y(t)$, $\dot{h}_x(t)$, $\dot{h}_y(t)$, and $h_a(t)$, the state of the block is given by $b_x(t)$, $b_y(t)$, $\dot{b}_x(t)$, $\dot{b}_y(t)$, and $b_a(t)$, and the relation between them is given by $c_x(t)$, $c_y(t)$, $\dot{c}_x(t)$, $\dot{c}_y(t)$, $e(t)$, and $\dot{e}(t)$. For each variable, Table 1 provides the physical range of values it can take on, the initial and final sets of landmarks, and the numerical value or range representing the agent's knowledge of the value of each landmark.

During learning if the block is knocked off the tray or if it is not moved for 300 timesteps, then it is put back on the tray in a random position within reach of the agent.

3.2 Experimental Results

We evaluate the algorithm using the simple task of moving the block in a specified direction. We ran the algorithm five times using active learning and five times using passive learning and each run lasted 120,000 timesteps. The landmarks learned during one of the active runs are shown in Table 1, and some example rules from that run are shown in Table 2. Each active run of the algorithm resulted in an average of 62 predictive rules.

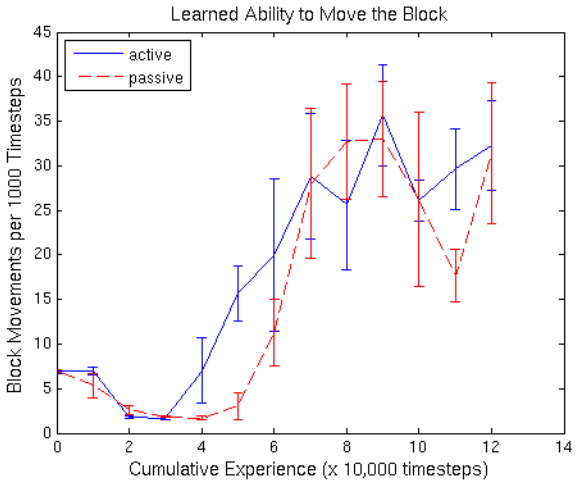


Figure 3: This graph shows the agent's performance on the task of hitting the block as a function of experience as it executes the learning algorithm of Section 2.7. The x axis represents the cumulative experience of the agent, and the y axis represents the average number of times the agent was able to hit the block within 1000 timesteps when the agent's current model was extracted and used for evaluation. Each data point represents the average of $n = 5$ runs and the error bars give the standard error.

Every 10,000 timesteps we evaluated each run on the task, and each data point in Fig. 3 represents the average of the five active runs or the five passive runs. The results in Fig. 3 show that using both passive and active exploration the agent gains proficiency as it learns until reaching threshold at approximately 70,000 timesteps. After reaching threshold performance there is high variance in the results due to the fluctuations in the statistics affecting the determinism of the rules that the agent uses to hit the block (we are currently working to remedy this). We also see in Fig. 3 that the agent learns more quickly using active exploration. For example, we see that the level reached by passive exploration after 60,000 timesteps is achieved by active exploration in just over 40,000 timesteps.

During each evaluation the agent undergoes 10 trials with each trial lasting 1,000 timesteps. During each trial a goal g is continually specified and the agent works until g is achieved or 50 timesteps pass, at which time the hand is moved back to its initial position and a new goal is specified. This process continues until the end of the trial, at which point the number of achieved goals is noted.

The specified goal g is either $\dot{b}_x \rightarrow (-\infty, 0)$, $\dot{b}_x \rightarrow (0, \infty)$, or $\dot{b}_y \rightarrow (0, \infty)$ based upon the relative positions of the hand and the block. To achieve g the agent continually forms a plan using the simple backchaining method described in Section 2.6. It first attempts to make a plan for g, associated with this plan is a probability of success calculated by taking the product of the state-dependent reliabilities of each rule during backchaining. If the plan for g does not have a probability of success above 0.1 then the agent creates a plan for each of the four goals $\dot{h}_x \rightarrow (-\infty, 0)$, $\dot{h}_x \rightarrow (0, \infty)$, $\dot{h}_y \rightarrow (-\infty, 0)$, and $\dot{h}_y \rightarrow (0, \infty)$ and chooses one of these randomly weighted by the reliability of its plan. If none of these goals has a reliability above 0.1 then the agent sets the plan to be a random action in the form of setting the motor variables $\tilde{u}_x$ and $\tilde{u}_y$ to random values with a termination criterion of ten timesteps. The agent then executes the plan. If the plan is terminated before g is achieved, then the agent creates and executes a new plan.

4. Space and Time Complexity

The number of possible events e is the sum of the number of qualitative values for each variable. The storage required to learn new rules is $O(e^2)$. The number of rules is $O(e^2)$, but only a small number are learned by the agent. Using marginal attribution each rule requires storage $O(e)$, although we store all pairs of events for simplicity. To learn landmarks the agent saves the last 20,000 timesteps, and for each rule that generates a landmark the agent saves the real values of each variable at the last 500 activations.

During exploration, at each timestep the algorithm examines and possibly updates statistics for each pair events for rule learning, and updates the statistics for each activated rule and event that occurs.

5. Conclusions and Future Work

We have presented a method that with the aid of active learning allows an agent to learn contingencies in its environment. At the beginning of the learning process the agent could only determine the direction of movement of an object, but by actively exploring its environment and using rules to learn new distinctions, and in turn using those distinctions to learn more rules, the agent has progressed from having a very simple representation towards a representation that is aligned with the natural "joints" of its environment. In future work we plan to move to a serial arm and to improve the formalism of the method to reduce the reliance on parameters and make the method more parsimonious.

Acknowledgements

This work has taken place in the Intelligent Robotics Lab at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Intelligent Robotics lab is supported in part by grants from the National Science Foundation (IIS-0413257, IIS-0713150, and IIS-0750011) and from the National Institutes of Health (EY016089). We would also like to acknowledge NSF grant EIA-0303609.

References

Cohen, L. B., Chaput, H. H., and Cashon, C. H. (2002). A constructivist model of infant cognition. *Cognitive Development*, 17:1323–1343.

DeCasper, A. J. and Carstens, A. (1981). Contingencies of stimulation: Effects of learning and emotions in neonates. *Infant Behavior and Development*, 4:19–35.

Drescher, G. L. (1991). *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, MA.

Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuousvalued attributes for classification learning. In *Thirteenth International Joint Conference on Articial Intelligence*, volume 2, pages 1022–1027.

Klein, J. (2003). Breve: a 3d environment for the simulation of decentralized systems and artificial life. In *Proceedings of the eighth international conference on Artificial life*, pages 329–334.

Kuipers, B. (1994). *Qualitative Reasoning*. The MIT Press, Cambridge, Massachusetts.

Miller, G. A., Galanter, E., and Pribram, K. H. (1960). *Plans and the Structure of Behavior*. Holt, Rinehart and Winston.

Mugan, J. and Kuipers, B. (2007). Learning to predict the effects of actions: Synergy between rules and landmarks. In *Proceedings of the 6th (IEEE) International Conference on Development and Learning*.

Nilsson, N. J. (1980). *Principles of Artificial Intelligence*. Tioga Publishing Company.

Piaget, J. (1952). *The Origins of Intelligence in Children*. Norton, New York.

Schmidhuber, J. (1991). Curious model-building control systems. In *Proc. International Joint Conference on Neural Networks*.

Watson, J. S. (2001). Contingency perception and misperception in infancy: Some potential implications for attachment. *Bulletin of the Menninger Clinic*, 65:296–320.