

Concepts From Time Series

Michael T. Rosenstein and Paul R. Cohen

Computer Science Department, LGRC

University of Massachusetts

Box 34610

Amherst, MA 01003-4610

{mtr, cohen}@cs.umass.edu

Abstract

This paper describes a way of extracting concepts from streams of sensor readings. In particular, we demonstrate the value of attractor reconstruction techniques for transforming time series into clusters of points. These clusters, in turn, represent perceptual categories with predictive value to the agent/environment system. We also discuss the relationship between categories and concepts, with particular emphasis on class membership and predictive inference.

Introduction

This research is part of an effort to explain how sensorimotor agents develop symbolic, conceptual thought, as every human child does. As in (Cohen *et al.* 1996) we are trying to “grow” an intelligent agent from minimal beginnings by having it interact with a complex environment. One problem with such projects is the transformation of streams of sensor data into symbolic concepts, cf. the symbol grounding problem (Harnad 1990). Hence, the focus of this paper is an unsupervised learning mechanism for extracting concepts from time series of sensor readings.

Concepts are abstractions of experience that confer a predictive ability for new situations. Moreover, for this project we assume a *predictive semantics* where the meaning of a concept is the predictions it makes. This working definition applies equally well to both objects and activities and depends upon a notion of *category*. For instance, just as one can form the category TOY for objects BALL and TOP, one can also create the category PLAY for activities BOUNCE and SPIN. From the *interactionist* perspective (Lakoff 1984; Johnson 1987), this correspondence is a natural one. Indeed, objects and activities seem to be duals — linked by sensorimotor experience — with a category of one connected to a category of the other.

Epistemically, a category is simply a collection of instances (of objects or activities) and a concept is a category plus its *entailments* (or consequences of category

membership). A similar definition follows from the influential work of Rosch involving categories and their abstractions, called *prototypes* (Rosch & Lloyd 1978). A prototype is best understood as a representative category member that may or may not correspond to any observed instance. (In fact, the latter case is true for the results in this paper.) Thus, whenever such abstractions can take the place of a category, one can think of a concept as a category prototype plus its *meaning* or predictive inferences.

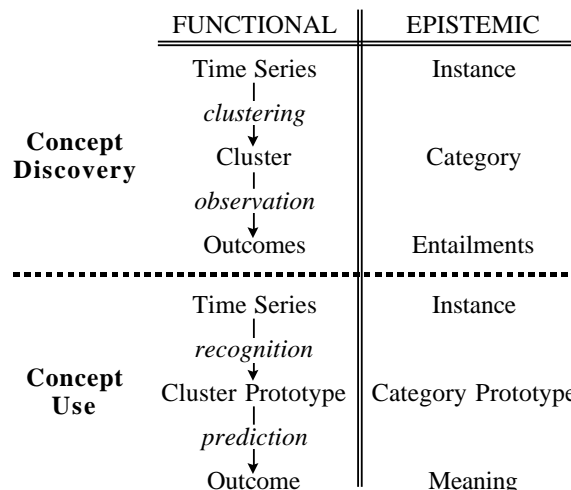


Figure 1: Functional view of concepts and the corresponding epistemic terms.

Figure 1 illustrates the way we realize these definitions of concepts. In particular, we begin with time series data and form clusters of points that have something in common. We then observe the subsequent outcomes, i.e., the future properties of the cluster members. Since we equate clusters with categories and outcomes with entailments, we also equate the corresponding acts of *clustering* and *observation* with the *discovery* of concepts. *Concept use* then follows a similar two-step procedure: (1) *Recognition*. Given a new time series, find the cluster prototype most like the new instance. (2) *Prediction*. Report the most likely outcome for the

cluster referred to by the matching prototype.

Experimental Environment

To demonstrate concept discovery and use, a simple experimental environment was created where two agents interact based on their predetermined movement strategies. These agents are entirely reactive and pursue or avoid one another using one of four basic behaviors:

1. NONE. The agent follows a line determined by its initial velocity. No attention is paid to the opponent's position.
2. AVOID. The agent attempts to move away from its opponent and avoid contact.
3. CRASH. The agent attempts to move toward its opponent and initiate contact.
4. KISS. The agent slows down before making contact (implemented as a combination of AVOID and CRASH).

Figure 2 shows two examples from the pursuit/avoidance simulator. During each trial, the agents interact only when they are close enough to detect one another, as represented by the inner circle in Figure 2. A trial ends when the agents make contact, when they get too far apart, or when the trial exceeds a time limit which is large compared to the duration of a typical interaction. The simulator implements a movement strategy by varying the agent's acceleration in response to the relative position of its opponent. (See (Rosenstein *et al.* 1997) for details.) In particular, movement strategies are equations of the form

$$a = i \cdot f(d, \dot{d}), \quad (1)$$

where a is acceleration, d is distance between agents, f is a function that gives one of the basic movement strategies, and i is a scale factor that represents the agent's strength or *intensity*.

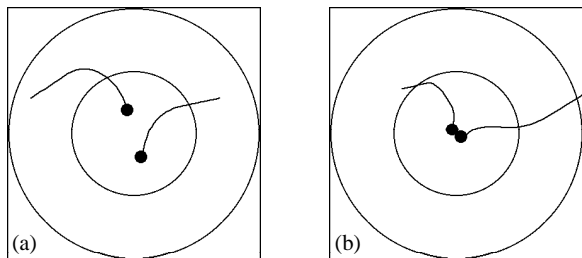


Figure 2: Simulator screen dump showing a representative trial of: (a) AVOID vs. CRASH; (b) KISS vs. KISS.

Activity Maps

Our previous approach to concept discovery was based on representations of dynamics called *activity maps* (Rosenstein *et al.* 1997). In keeping with the functional view of concepts in Figure 1, we also made the distinction between two types of activity maps: *behavior maps* for recognizing agent behaviors such as

CRASH or KISS, and *interaction maps* for predicting outcomes such as CONTACT or NO-CONTACT. These representations, in turn, were based on *phase portraits* and *basins of attraction* — common tools used by dynamicists for understanding system behavior.

During a learning phase, a library of activity maps was constructed by running thousands of trials in the simulator while recording the movement patterns of each agent as well as the outcome of every trial. In a supervised manner, one behavior map was built for each of eight agent types (four basic movement strategies with three levels of intensity for AVOID and CRASH), and one interaction map was built for each of the 36 distinct pairs of behaviors (64 possible pairs minus 28 symmetrically equivalent pairs). With the pursuit/avoidance simulator, this library of activity maps proved sufficient for recognizing the participants of a new trial and for predicting a CONTACT or NO-CONTACT outcome.

Table 1 illustrates the performance of the recognition algorithm in (Rosenstein *et al.* 1997). Interestingly, this *confusion table* demonstrates the misinterpretation of the various behaviors. For example, 66% of the time the algorithm confused KISS with one of the CRASH movement strategies, yet rarely mistook KISS for one of the AVOID types. The reason for this sort of confusion is that *apparent* behavior is dependent upon not only the agent's predetermined movement strategy, but also the circumstances, i.e., initial velocity and opponent behavior. Actually, in some situations, a KISS agent reacts just like a CRASH type, and vice versa. These results suggest that another way to categorize interactions is by the nature of the interaction itself. In fact, the explicit step of behavior recognition is no longer necessary with the clustering approach described shortly.

Actual	Recognizer Response							
	N	A-	A	A+	C-	C	C+	K
N	40	21	7	3	26	3	0	0
A-	18	53	19	5	5	0	0	0
A	2	16	74	8	0	0	0	0
A+	0	2	8	90	0	0	0	0
C-	26	6	3	1	35	23	0	6
C	2	2	0	1	21	25	21	28
C+	0	0	0	0	1	9	77	13
K	8	1	0	0	21	20	25	25

Table 1: Confusion table illustrating recognition performance with agent behaviors chosen randomly. Shown are response percentages, where behavior names are shortened to first letters only, and - and + indicate weak and strong forms, respectively.

Despite the success of activity maps at recognizing behaviors and predicting outcomes, the approach has two drawbacks worth mentioning. First, even for a simple simulator with just two agents, the size of the map library scales as $O(n^2)$, where n is the number of movement strategies. Preferably, the concept library should have size proportional to the number of needed concepts. Second, thousands of trials are necessary to build just one map and the associated learning algorithm must operate in a supervised fashion. Instead, an agent should find the relevant categories for itself, without the imposed biases of an external teacher. For these reasons, we developed the current approach to concept discovery based on clustering techniques. Below, we show that few clusters and few experiences are needed to recognize situations and predict outcomes — without a supervisor — and to do so quite accurately.

The Method of Delays

The formation of clusters requires a metric and so one must first devise a suitable metric space for the data. In this work we make use of an *attractor reconstruction* technique called the *method of delays* or *delay-space embedding*. Takens proved that the method of delays provides a means for mapping a time series to a topologically equivalent spatial representation (an *embedding*) of an underlying dynamical system (Takens 1981). This mapping is accomplished by forming an m -dimensional vector of *delay coordinates*,

$$\mathbf{X}_i^T = [x_i \ x_{i-J} \ x_{i-2J} \ \dots \ x_{i-(m-1)J}], \quad (2)$$

where $\{x_1, x_2, \dots, x_n\}$ is the n -point time series, m is called the *embedding dimension*, and J is the *embedding delay*. As described in (Rosenstein, Collins, & De Luca 1994), and references therein, the practical application of Takens theorem requires some care on the experimenter’s part when choosing the delay value. In any case, the theoretical basis of attractor reconstruction is geared toward dynamical systems, although the techniques often prove useful for uncovering patterns in time series data, regardless of their source.

For instance, Figure 3 shows a *delay portrait* with representative trajectories leading to three different outcomes in the pursuit/avoidance simulator. Each curve is based on the distance time series recorded during one interaction. We chose *object-distance* as the agent’s only sensor because we want to demonstrate concept discovery while supplying as little innate knowledge or structure as possible. Additionally, the ability to measure object distance is a reasonable assumption since most mobile robots are equipped with sonar or video-based position sensors.

The method of delays is important for this research, not because we have attractors to reconstruct, but because we need some basis for discovering concepts without detailed knowledge of the agent/environment system. In all but the simplest applications, an agent is unable to measure all relevant quantities in the world, and so the intuition behind delay-space embeddings is

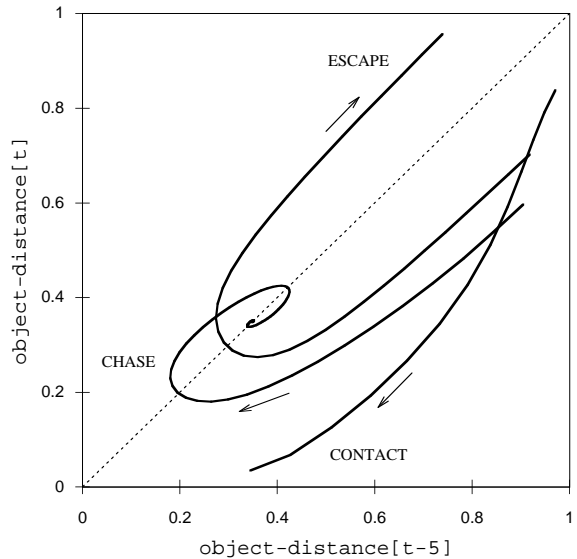


Figure 3: Two-dimensional delay portrait with $J = 5$ and trajectories illustrating CONTACT, ESCAPE, and CHASE.

this: if the state of the environment at time t is uncertain because the agent has, say, one sensor, then examination of sensor readings prior to time t will reduce the ambiguity.

Concepts and Clusters

As shown earlier in Figure 1, there is a close relationship between concepts and categories. Indeed, “there is nothing more basic than categorization to our thought, perception, action, and speech” (Lakoff 1984). But before any agent can make use of concepts, it must first discover the pertinent categories.

One general technique for discovering categories is to form clusters of points in a suitable space. This was the basis of Elman’s work on learning lexical classes from word co-occurrence statistics (Elman 1990). Elman first trained a recurrent neural network to predict successive words in a long input string. This then set the stage for hierarchical clustering of the hidden-unit activation space, where the result was groups of words that coincide with classes like NOUN-FOOD or VERB-PERCEPT. Similarly, Omlin and Giles described a way to identify clusters in a network’s hidden-unit space, with each cluster representing a node in an associated finite-state machine (Omlin & Giles 1996).

Our approach to clustering works directly from delay coordinates and proceeds with no prior training. For the pursuit/avoidance simulator, we begin with the first observed interaction and immediately create a cluster consisting of this lone experience. A cluster itself is simply a data structure that stores the number of members, the frequency of each outcome, and a prototype for the class. Whenever a new experience arrives, the algorithm creates a new cluster from the data and then

attempts to merge existing clusters based on a measure of similarity. Specifically, two clusters are replaced by one formed from their constituent information whenever the Euclidean distance between them (in delay-coordinate space) is less than a threshold parameter, θ_s . Hence, the algorithm continually updates its list of categories to reflect new experiences. Central to this updating procedure is the use of cluster prototypes.

Figure 4 shows the six cluster prototypes derived from 100 agent interactions, where the movement strategies were chosen randomly and the similarity threshold was set to 20% of the range in the distance data. Each prototype is simply the average object-distance time series formed from all interactions that make up a given cluster. (Strictly speaking, a prototype consists of m delay coordinates, although we show the entire time series to illustrate distinctive patterns.) This implementation is entirely consistent with Rosch’s work on prototypes (Rosch & Lloyd 1978), and also serves two practical purposes: as a way for testing cluster similarity, and as a way for understanding the meaning of a cluster.

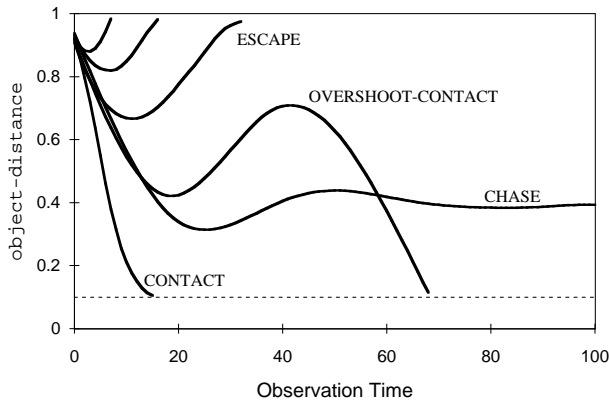


Figure 4: Cluster prototypes formed at observation time $t = 50$, with similarity threshold $\theta_s = 0.2$, embedding dimension $m = 5$, and embedding delay $J = 1$.

The prototypes in Figure 4 correspond to six different categories of agent interactions, with each category possessing its own entailments. Thus, the clustering algorithm discovered six concepts about the experimental domain. Moreover, these prototypes reflect actual differences in the simulated environment. In particular, the algorithm found concepts that one could describe as “chase,” “contact,” contact after the agents first “overshoot” one another, and “escape” with short, medium, and long escape times. But are these clusters any good for recognition and prediction?

The agents in the pursuit/avoidance simulator have implicit goals of CONTACT and ESCAPE. These goals correspond to two possible outcomes, and the experimental domain affords a third, emergent outcome of CHASE. To evaluate the usefulness of a set of clusters, we generated 1000 additional interactions, predicted

one of these outcomes for each trial, and recorded the percentage of correct responses. The prediction scheme was a straightforward voting algorithm that followed the two-step procedure for concept use: (1) *Recognition*. Find the cluster prototype nearest the time series in delay-coordinate space. (2) *Prediction*. Report the majority outcome for the corresponding cluster. Figure 5 illustrates the prediction performance as a function of observation time (the time when clusters are formed). The graph shows that the algorithm’s response improves as the interaction unfolds and more data become available. For example at observation time $t = 10$ the prediction is correct only 64% of the time, but by $t = 35$ prediction performance exceeds 90%.

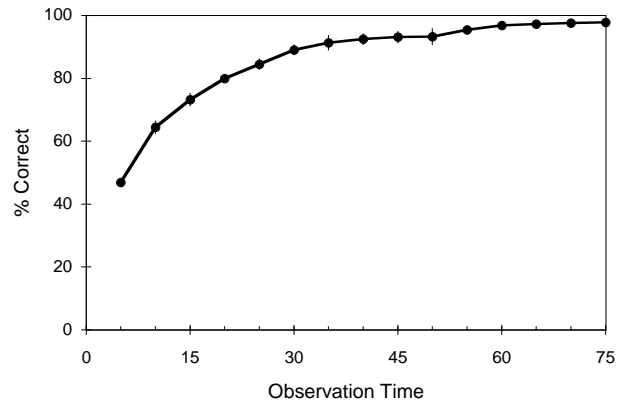


Figure 5: Prediction performance versus observation time for 1000 test interactions and the clusters shown in Figure 4. Clustering was based on 100 trials with $\theta_s = 0.2$, and error bars show the standard deviation for five replications.

How Many Concepts?

Our previous work using three intensity levels, $i = \{0.5, 1.0, 2.0\}$, required 36 interaction maps and, therefore, 36 concepts. With our present effort, we made things more difficult and selected i randomly from the interval $[0.5, 2.0]$. Nevertheless, clustering in delay-coordinate space resulted in far fewer concepts with no substantial differences in prediction performance. Even when we increased the number of trials to 10,000, we observed little change in the number of clusters formed. (See Figure 6.) In fact, very few trials — and, therefore, very few clusters — were needed to achieve a reasonable level of performance, after which the algorithm improved prediction performance by fine-tuning the existing clusters rather than creating new ones. Our explanation is that the algorithm discovered all the concepts that were available for discovering in the first place.

One caveat about clustering is that the results can become skewed by making a poor choice for the similarity threshold. For example, Figure 7a shows the number of clusters formed for several values of θ_s . When the similarity threshold is too small, trials seem to have little in common with one another and the number of clus-

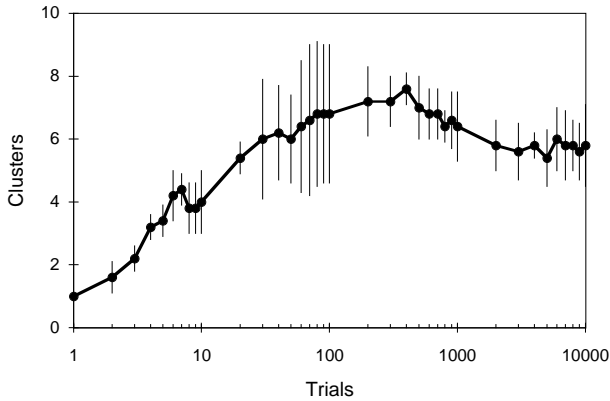


Figure 6: Semi-log plot of the number of clusters formed versus number of trials, where $\theta_s = 0.2$. Error bars show the standard deviation for five replications.

ters is roughly the same as the number of trials. As expected, an increase in θ_s yields a shorter cluster list, with the total number approaching the limiting value of 1. Figure 7b illustrates the other half of the story. When θ_s is too large, too few concepts are discovered and prediction performance is poor. But once the number of clusters exceeds a value of about five, prediction performance saturates and there is no benefit to learning a more detailed breakdown of “concept space.” Put differently, plots such as Figure 7 offer a way to determine the number of concepts worth learning in the given domain.

Delay Coordinates

Earlier, we characterized the method of delays as a technique for reducing ambiguity in sensor readings. To demonstrate this benefit of delay coordinates we performed clustering, not at a particular observation time, but instead at a given value for `object-distance`. In other words, we fixed the first delay coordinate so all trials appear the same to a naive algorithm that clusters along just one dimension. Moreover, we made use of delay coordinates *forward* in time, with Eq. (2) replaced by

$$\mathbf{X}_i^T = [x_i \ x_{i+J} \ x_{i+2J} \ \dots \ x_{i+(m-1)J}]. \quad (3)$$

In practice, delay coordinates — both forward and backward in time — require the agent to wait until the data become available. The difference is simply a matter of perspective. For the forward view, the agent buffers its sensor readings when triggered by an event, e.g., the proximity of an opponent. For delay coordinates backward in time, the agent updates its sensor buffer continuously and associates an event with the most recent delay coordinate, rather than the oldest one.

Figure 8a shows the result of clustering in a five-dimensional delay-coordinate space with delay $J = 1$. One prototype is similar to the category for `CONTACT` in

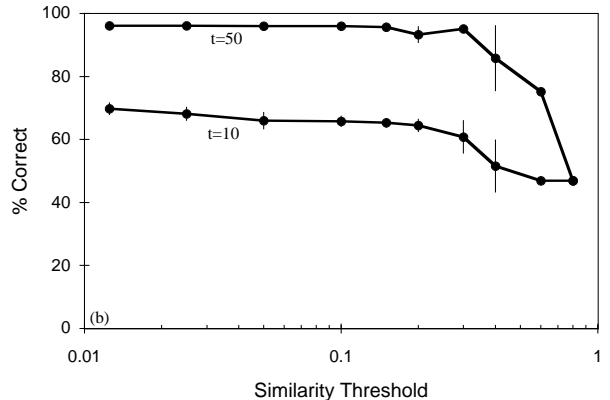
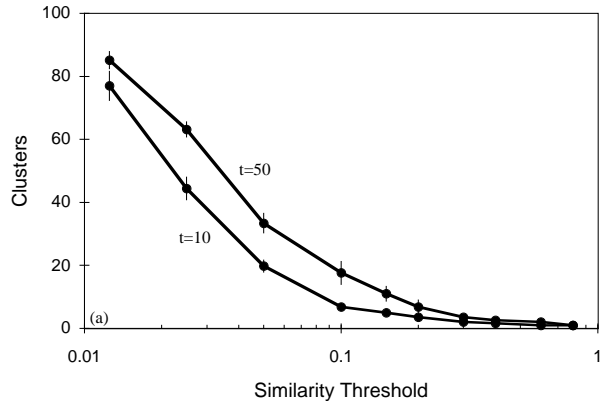


Figure 7: Effects of similarity threshold on (a) number of clusters and (b) prediction performance, for observation times $t = 10, 50$. Clustering was based on 100 trials and error bars show the standard deviation for five replications.

Figure 4, whereas the other is an amalgam of all three basic outcomes. For Figure 8b, we increased the embedding delay to $J = 3$ so the delay-coordinate vector in Eq. (3) spans a larger portion of the sensor stream for the same fixed dimension. The effect is clusters of greater homogeneity since the algorithm is able to capture many of the subtle distinctions between two interactions.

Figure 9 illustrates more clearly the role of the embedding dimension. When m is too small, there is little time to observe a change between the first and last delay coordinates, and so many interactions fall near one another in delay-coordinate space. In this case, few clusters form and the prediction performance is poor (much like Figure 7 when the similarity threshold is too large). As m increases, points spread apart in embedding space since the additional coordinates reveal latent differences in the corresponding time series.

Interaction Maps

With the pursuit/avoidance simulator, the clustering approach to concept acquisition afforded a prediction performance exceeding 95%. However, in more compli-

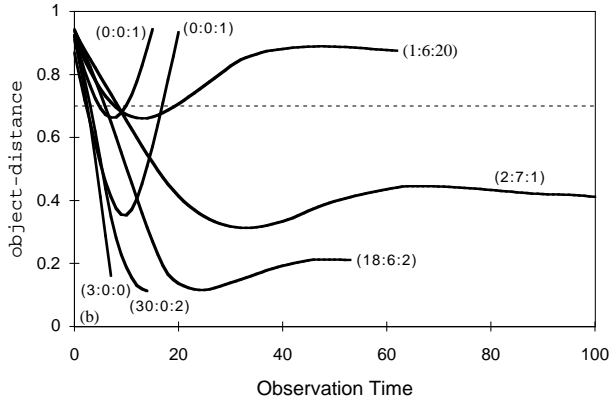
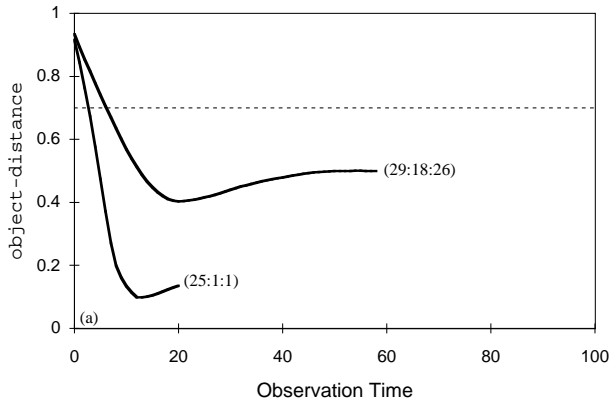


Figure 8: Cluster prototypes for **object-distance**=0.7, with $\theta_s = 0.1$, embedding dimension $m = 5$, and embedding delay (a) $J = 1$ and (b) $J = 3$. Labels show the makeup of each cluster. For instance, (29:18:26) indicates that the prototype was formed from 29 interactions ending in CONTACT, 18 ending in CHASE, and 26 ending in ESCAPE.

cated, possibly noisy, environments it may not be possible to achieve this level of success with a prediction algorithm based solely on outcome frequencies. Our solution is to exploit the dynamics in the environment and store more detailed information about entailments in the form of interaction maps. We avoid the combinatorial drawbacks of map libraries by creating just one interaction map per cluster.

Interaction maps are similar to diagrams that show *basins of attraction*, or sets of initial conditions that lead to different *limit sets* (long-term outcomes). One way to build an interaction map involves the partitioning of delay-coordinate space into cells, where every cell contains a counter for each interesting outcome. As two agents interact, they generate a trajectory in this space, and once the outcome is known, the corresponding counter is incremented in each cell visited by the trajectory. The map is then colored as a function of the counter values in each cell.

Figure 10 shows the interaction map for the extreme case where all trials are placed into just one cluster.

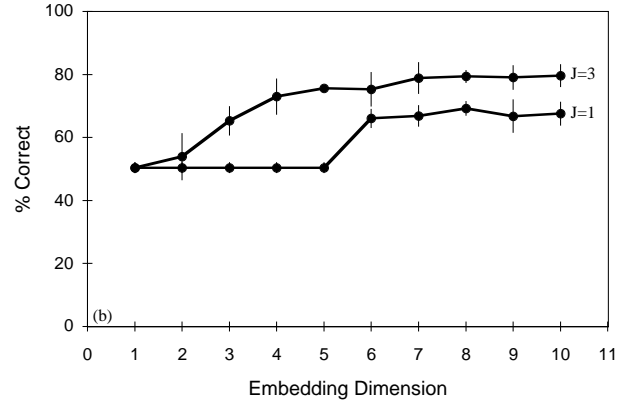
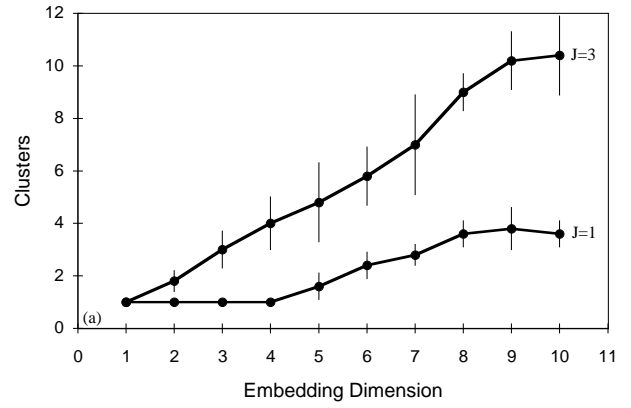


Figure 9: Effects of embedding dimension on (a) number of clusters and (b) prediction performance, for embedding delays $J = 1, 3$. Clustering was based on 100 trials, where $\theta_s = 0.1$ and **object-distance**=0.7. Error bars show the standard deviation for five replications.

(For graphical purposes, we split the map into three grayscale images, one for each outcome, although one color diagram is often more informative.) Whenever a trajectory enters a dark region of a map, one can confidently predict the corresponding outcome. With interaction maps we saw a boost in prediction performance from 48% to 78% at $t = 10$ and to 88% at $t = 50$. Notice that these results compare favorably to those for six clusters as in Figure 5.

Conclusions

Perhaps the most promising aspect of this work is the possibility of an agent discovering concepts for itself. An emphasis on the agent not only aids our understanding of human cognitive development, but also influences the design of autonomous agents for more pragmatic purposes. In practical applications, someone, either agent or designer, must carve up the world in some appropriate way, but what one deems relevant is quite dependent on one's perceptions of the world and one's ability to affect those percepts through action. For instance, as part of another project in our lab (Schmill *et*

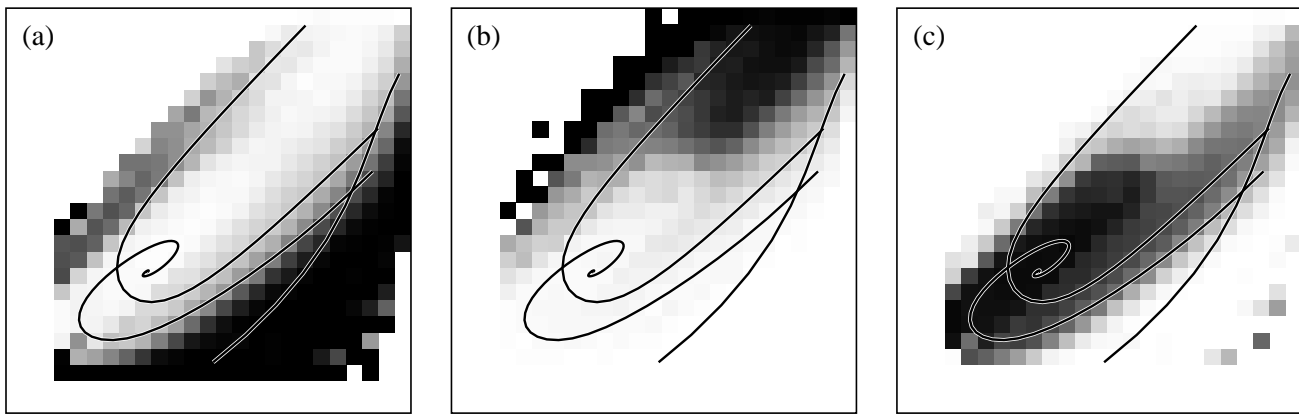


Figure 10: Interaction maps from one cluster with outcomes (a) CONTACT, (b) ESCAPE, and (c) CHASE. Darker shades of gray indicate a greater likelihood of the associated outcome. Also shown is an overlay of the delay portrait in Figure 3.

al. 1998), a mobile robot discovered an unexpected solution to the problem of being jammed between two objects. Whereas a programmer might instruct the robot to rock back-and-forth — as an automobile driver caught in mud or snow — the robot found on its own that opening its gripper could push itself far enough away from one of the objects. The point of this example is that a solution was found through interaction with the world. Indeed, Lakoff (Lakoff 1984) and Johnson (Johnson 1987) provide cogent arguments that more advanced symbolic thought is founded upon this very sort of sensorimotor interaction.

Although it may be a bit of a stretch to say that our pursuit/avoidance agents perform “sensorimotor” interaction, the results in this paper provide a beginning for future work with mobile robots. At the heart of the approach is the discovery of sensor reading patterns, and the specific contribution of this paper is a technique for deducing such patterns, i.e., clusters, from time series data. These clusters, together with their entailments, then provide a means for recognizing situations and predicting outcomes in the agent’s world.

Acknowledgments

This research is supported by the National Defense Science and Engineering Graduate Fellowship and by DARPA under contract No. N66001-96-C-8504. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

Cohen, P. R.; Oates, T.; Atkin, M. S.; and Beal, C. R. 1996. Building a baby. In *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, 518–522.

Elman, J. L. 1990. Finding structure in time. *Cognitive Science* 14:179–211.

Harnad, S. 1990. The symbol grounding problem. *Physica D* 42:335–346.

Johnson, M. 1987. *The Body in the Mind*. University of Chicago Press.

Lakoff, G. 1984. *Women, Fire, and Dangerous Things*. University of Chicago Press.

Omlin, C. W., and Giles, C. L. 1996. Extraction of rules from discrete-time recurrent neural networks. *Neural Networks* 9(1):41–52.

Rosch, E., and Lloyd, B. B. 1978. *Cognition and Categorization*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Rosenstein, M.; Cohen, P. R.; Schmill, M. D.; and Atkin, M. S. 1997. Action representation, prediction and concepts. University of Massachusetts Computer Science Department Technical Report 97-31, also presented at the 1997 AAAI Workshop on Robots, Softbots, Immobots: Theories of Action, Planning and Control.

Rosenstein, M. T.; Collins, J. J.; and De Luca, C. J. 1994. Reconstruction expansion as a geometry-based framework for choosing proper delay times. *Physica D* 73:82–98.

Schmill, M. D.; Rosenstein, M. T.; Cohen, P. R.; and Utgoff, P. 1998. Learning what is relevant to the effects of actions for a mobile robot. To appear in *Proceedings of the Second International Conference on Autonomous Agents*.

Takens, F. 1981. Detecting strange attractors in turbulence. *Lecture Notes in Mathematics* 898:366–381.