Growing Ontologies

Paul R. Cohen
Department of Computer Science
University of Massachusetts, Amherst, MA 01003
cohen@cs.umass.edu
Tracking number: A660
Content areas: ontologies, machine learning, philosophical foundations, robotics, software agents
Submitted to IJCAI-99

Abstract

Conceptual structures or ontologies are usually built by hand by skilled knowledge engineers. This paper presents a theory of how ontologies may be learned by intelligent agents interacting with their environment in an unsupervised way. Categories of activities are learned, then abstractions over those categories result in concepts. The entire ontology is based on activities. The meanings of concepts are discussed. Systems that learn categories of activities and concepts are presented and their general characteristics are described.

Statement of sole submission: The author certifies that this paper has not been accepted by and is not currently under review for another conference or journal. Nor will it be submitted to such during IJCAI-99's review period.

Growing Ontologies

Tracking number: A660
Content areas: Philosophical foundations, ontologies, cognitive modeling, commonsense, unsupervised learning

Abstract

Conceptual structures or ontologies are usually built by hand by skilled knowledge engineers. This paper presents a theory of how ontologies may be learned by intelligent agents interacting with their environment in an unsupervised way. Categories of activities are learned, then abstractions over those categories result in concepts. The entire ontology is based on activities. The meanings of concepts are discussed. Systems that learn categories of activities and concepts are presented and their general characteristics are described.

Introduction

The subject of this paper is the foundation of conceptual systems. In artificial intelligence, conceptual systems are sometimes called ontologies and they are usually built by highly-trained knowledge engineers. It's less clear how humans acquire conceptual systems, or rather, the scope of the innate endowment is unclear. In any case, if machines could acquire conceptual knowledge with the same facility as humans, then AI would be much better off. Many people think that our conceptual system is a prerequisite for all sorts of intelligent processes, from analogical reasoning to natural language understanding, from reasoning under uncertainty to computer-assisted cooperative work. Lenat and Feigenbaum (1987) promised us that a sufficiently large corpus of common sense knowledge would "go critical" and start learning, by reading, autonomously. That hasn't happened yet, but there's no denying the dream of a machine that knows roughly what we know, organized roughly as we organize it, with roughly the same values and motives as we have.

It makes sense, then, to ask how this knowledge is acquired by humans and how might it be acquired by machines. In particular and for various reasons, I want to focus on the origins of conceptual knowledge, the earliest distinctions and classes, the first efforts to carve the world at its joints. One reason is just the scientist's pleasure at getting to the bottom of, or in this case the beginning of, anything. Another reason is that the origin of conceptual systems is currently hotly debated: Some people think that neonates are born with moderately sophisticated conceptual systems (e.g., Baillargeon, 1994; Carey and Gelman, 1991; Spelke et al. 1992), others dispute this and seek an empiricist, or non-nativist, account of development (e.g., Mandler, 1988, 1992). I think one has to take a minimalist stance and avoid innate knowledge in one’s explanations of the acquisition of later knowledge. Partly this reluctance comes from years of slogging in AI, where it seems we must always provide a lot of knowledge for our systems to do relatively little with. I want to focus on the first concepts because unless I do, I will have to provide them by hand, which is a bore, and also makes me very uncertain about the explanatory power of what follows. If instead we ask how does the machine come to conceptualize the world this way rather than that, if we ask the machine to form its own conceptual system, then we don't have to do the hard work and our claims to understanding intelligence are that much stronger.

The principal claim of this paper is that *concepts can be learned without supervision by abstracting over representations of activities*. Piaget (1952) insisted that concepts must arise out of activities — because simple action schemas develop before conceptual thought — but the mechanisms he proposed to explain concept acquisition were vague by the standards of artificial intelligence. My strategy in this paper is to present methods for unsupervised learning of clusters of activities, implemented in robots and software agents, and argue that these representations identify categories and concepts.

Why Should Agents Learn Concepts?

What are concepts and ontologies for and why should they be learned rather than constructed by knowledge engineers? Imagine yourself to be a mobile robot wandering around the lab. What concepts do you need, and what will you do with them? Presumably you have some reactive routines to keep you out of trouble. These recognize aspects of your internal state and the world state, and generate actions in response; for example, when you detect an obstacle in your path, you change direction or stop. Suppose you have a function `obstacle-in-path` that takes as input some sensory information such as sonar readings. As described, the concept "obstacle" simply doesn't exist for you, the robot. It exists for the person who wrote the function `obstacle-in-path`, but not for you. You do not know that obstacles may sometimes be pushed aside, although you may have a function to push obstacles aside; you do not know that obstacles impede paths, although obstacles impede your paths; you do not even know what a path is, although you trace one whenever you move. You have no conceptual structure whatsoever, just a bunch of routines to keep out of trouble. And why *should* you know anything, if these routines work? Why do you need concepts like "obstacle" and "impede" and "path" at all?

The function `obstacle-in-path` was written by a person who thought about all the situations you might encounter and realized that some of them involve an “obstacle” in your “path.” Because this person

conceptualized your experiences, you have an appropriate response to a common situation. Your behavior is organized around your programmer's conceptual structure. This structure — of which you are innocent — serves you well. This is what concepts are for: They give you interpretations of your sensors, a basis for judgments that situations are similar, and the distinctions on which you decide what to do. Concepts are necessary even when they are not explicit. We must stop deluding ourselves that a robot or any other agent is capable of intelligent behavior without a conceptual structure. If `obstacle-in-path` produces intelligent behavior, the intelligence must be attributed to the programmer who dreamed up the concepts "obstacle" and "path." It's delusional to say, "Our robot achieved so much with nothing more than a handful of reactive routines"; the reactive routines are just the business end of an entirely hidden but sophisticated conceptual system.

From Activities to Concepts

Here is the problem to be solved: An agent such as a robot is "born" with a small set of physical activities but no conceptual system. As it interacts with its environment, it forms categories, and by induction over category instances, concepts. How does this work? What innate structures are required? This section presents some implemented methods that solve parts of the problem.

How the Pioneer 1 Robot Learns Prototypes

The Pioneer 1 robot has two independent drive wheels, a trailing caster, a two degree of freedom gripper, and roughly forty sensors including five forward-pointing sonars, two side-pointing sonars, a rudimentary vision system, bump and stall sensors, and sensors that report the state of the gripper. The robot is controlled by a remote computer, connected by radio modem.

The robot's state is polled every 100msec., so a vector of 40 sensed values is collected ten times each second. These vectors are ordered by time to yield a multivariate time series. Figure 1 shows four seconds of data from just four of the Pioneer's forty sensed values. Given a little practice, one can see that this short time series represents the robot moving past an object. Prior to moving, the robot establishes a coordinate frame with an x axis perpendicular to its heading and a y axis parallel to its heading. As it begins to move, the robot measures its location in this coordinate frame. Note that the ROBOT-X line is almost constant. This means that the robot did not change its location on a line perpendicular to its heading, that is, it did not change its heading, during its move. In contrast, the ROBOT-Y line increases, indicating that the robot does increase its distance along a line parallel to its original heading. Note especially the VIS-A-X and VIS-A-Y lines, which represent the horizontal and vertical locations, respectively, of the centroid of a patch of light on the robot's "retina," a CCD camera. VIS-A-X decreases, meaning that the object drifts to the left on the retina, while VIS-A-Y increases, meaning the object moves toward the top of the retina. Simultaneously, both series jump to constant values. These values are returned by the vision system when nothing is in the field of view. In sum, the four-variable time series in Figure 1 represents the robot moving in a straight line past an object on its left, which is visible for roughly 1.8 seconds and then disappears from the visual field.

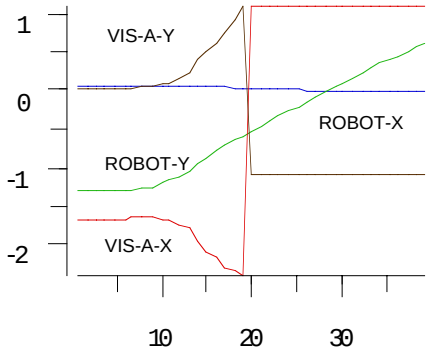

Figure 1. As the robot moves, an object approaches the periphery of its field of view then passes out of sight.

Every time series that corresponds to moving past an object has qualitatively the same structure as the one in Figure 1: ROBOT-Y increases; VIS-A-Y increases to a maximum then takes a constant value; and VIS-A-X either increases or decreases to a maximum or minimum depending on whether the object is on the robot's left or right, then takes a constant value. ROBOT-X might change or not, depending on whether the robot changes its heading or not.

It follows that if we had a statistical technique to group the robot's experiences by the characteristic patterns in multivariate time series (where the variables represent sensor readings), then this technique would in effect learn a taxonomy of the robot's experiences. *Clustering by dynamics* is such a technique. The version we describe here was developed by [name removed for blind review] ; similar methods are described in [references removed for blind review] :

1. A long multivariate time series is divided into segments, each of which represents an *episode* such as moving toward an object, avoiding an object, crashing into an object, and so on. Episode boundaries can be inserted by humans or by a simple algorithm that looks for simultaneous changes in multiple state variables. Obviously we prefer the latter technique (and apply it in [references removed for blind review]) because it minimizes human involvement in the learning process; however, for the experiment described here, episode boundaries were marked by us. We did *not* label the

episodes in any way.

2. A *dynamic time warping* algorithm compares every pair of episodes and returns a number that represents the degree of similarity between the time series in the pair. Dynamic time warping is a technique for "morphing" one multivariate time series into another by stretching and compressing the horizontal (temporal) axis of one series relative to the other (Sankoff and Kruskal, 1983). If two multivariate series are very similar, relatively little stretching and compressing is required to warp one series into the other. A number that indicates the amount of stretching and compressing is thus a proxy for the similarity of two series.
3. Having found this similarity number for the series that correspond to every pair of episodes, it is straightforward to cluster episodes by their similarity. Agglomerative clustering is a method to group episodes by similarity such that the within-cluster similarity among episodes is high and the between-cluster similarity is low.
4. Another algorithm finds the "central member" of each cluster, which we call the cluster *prototype* following Rosch (1975).

In a recent experiment, this procedure produced prototypes corresponding to passing an object on the left, passing an object on the right, driving toward an object, bumping into an object, and backing away from an object [reference removed for blind review].

These prototypes were learned largely without supervision and constitute a primitive ontology of activities – the robot learned some of the activities it can do in its environment. What supervision or help did we provide? We wrote the programs that controlled the robot and made it do things. We divided the time series into episodes (although this can be done automatically). We limited the number of variables that the dynamic time warping code had to deal with, as it cannot efficiently handle multivariate series of forty state variables. We did not label the episodes to tell the learning algorithm which clusters of activities it should consider. In fact, the only guidance we provided to the formation of clusters was a threshold statistic for adding an episode to a cluster. To reiterate, we did not anticipate, hope for, or otherwise coerce the algorithms to learn *particular* clusters and prototypes. The robot's ontology of activities is its own.

Prototypes from Delay Coordinate Embeddings

To cluster episodes, one needs to judge their similarity. This function was accomplished by dynamic time warping in the previous study. Another way to do it is with delay coordinate embedding [reference removed for blind review]. Let an episode $e_i = X_{i,t-\tau}, \mathsf{K}, X_{i,t-1}, X_{i,t}$ be a time series of sensor X for $\tau+1$ time steps ending with an event at time t. Clearly, e_i can be represented by a single point p_i in a space of $\tau+1$-dimensions, as each of $X_{i,t-\tau}, \mathsf{K}, X_{i,t-1}, X_{i,t}$ identifies the location of p_i in one dimension. This transformation of a $\tau+1$-step time series to a point in $\tau+1$-dimensional space is called a delay coordinate embedding. The Euclidian distance between two points p_i and p_j is a measure of similarity of episodes e_i and e_j.

In one experiment [reference removed for blind review] we implemented two simulated agents, each of which adopts one of nine behaviors, including crash, avoid, kiss, and so on. For instance, A might try to *avoid* B while B tries to *crash* into A. We ran several episodes of pairs of behaviors. Each episode was represented by a time series of the distance d between the agents. An episode ended when the agents touched each other or the distance between the agents exceeded a threshold. Delay coordinate embeddings were constructed for each episode, the Euclidean distance $p_i - p_j$ was calculated for each pair of episodes e_i, e_j, and these distances were used as a similarity metric for agglomerative clustering of episodes.

With very little training, this procedure came up with six clusters. The prototypes of these clusters, obtained by averaging the time series within each cluster, are shown in Figure 2. Prototypes 1, 2 and 3 involve one agent eluding the other; the difference between the clusters is how close the agents get to each other before moving apart. Prototype 4 represents interactions in which the agents repeatedly draw together and move apart over time, but never touch – a kind of perpetual chase. Prototype 5 is a case in which the agents close on each other so quickly that inertia carries them past each other – they overshoot one another – then they change direction and eventually touch. Prototype 6 is simple case in which the agents rapidly draw together and touch.

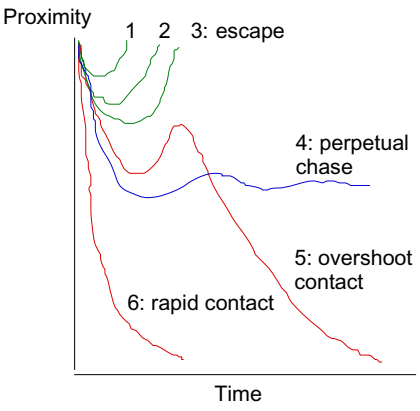

Figure 2. Six prototypes for agent interactions, learned without supervision, and plotted as time series of the proximity between the agents.

Our clustering system was not instructed to cluster episodes by their outcomes (e.g., touch, escape, etc.); indeed, the clusters and their prototypes were constructed entirely without supervision. But time series are so redundant that clustering by dynamics often produces prototypes that have qualitatively different outcomes. As in the previous study, we claim that the prototypes learned by delay coordinate embedding constitute a primitive ontology of activities. Because the learning is unsupervised, the ontology is not ours, but is acquired through experience. In fact, we were quite surprised to see the "overshoot contact" prototype. It is an emergent interaction of the agents that we had never noticed.

Learning an Ontology of Objects

Coelho and Grupen (1997) have shown how the dynamics of grasping objects are sufficient to identify classes of objects. A robot hand attempts to grasp prismatic objects by activating controllers that run to convergence, attempting to minimize force and wrench residuals. The net effect of running these controllers is that the fingers of the hand migrate over the surface of the object until they achieve a stable grasp. These migrations describe trajectories in the two-dimensional error (residual) space. (In the terms of the previous studies, each grasping episode is represented by a time series of force and wrench residuals.) The union of all trajectories for a given object is called a preimage, and the shape of the preimage depends on the object geometry. Learning preimages is supervised — the algorithm knows what kind of block it is trying to pick up — but the set of preimages constitutes a primitive ontology that supports classification. The robot can identify objects by feel as it attempts to gain a stable grasp around them. This is a very clear example of a conceptual activity — classifying objects —that arises out of a purely sensorimotor activity. In fact, when the hand conceptualizes an interaction with an unidentified block as, say, grasping a hexagonal prism, it can predict instabilities in the grasp (i.e., entailments) and modify its grasp.

Learning Grammatical Categories from Word Sequence Dynamics

Elman (1995) trained a recurrent neural network to predict the next word in sentences, then clustered words by their hidden unit representations. Elman argues that because the network was recurrent, the hidden units represent the dynamics of sentences, that is, the transitions between words. Remarkably, the clusters of words appear to correspond to interesting grammatical categories such as direct object, verb, and so on. Elman's approach is supervised in the sense that the network gets feedback on what it is trying to learn, but unsupervised in the more important sense of inducing categories of words without being told which categories the words belong to.

Learning Prototypes by Clustering by Dynamics

The four preceding examples, two from our lab and two developed elsewhere, are instances of *clustering by dynamics*: Starting with time series representations of episodes (in Elman's case, time series of words), cluster similar time series, then average the elements in a cluster to get a prototype. Although this method is extremely simple, it appears sufficient to learn concepts given unclassified instances of activities, that is, without supervision. I claim that prototypes are concepts, but in what sense, and what do these concepts mean? I turn to these questions now.

Concepts, Conceptualizations and Meaning

Definitions of *concept* differ but I believe all have in common that concepts are abstractions. Often, concepts are taken to be abstractions over features or attributes of objects. I want focus not on the attributes of objects but on the dynamics of activities. Thus I define concepts as *abstract representations of activities (i.e., prototypes) plus the participants in and the entailments of the activities*. Concepts for objects and other participants in activities will be learned (as they were by Coelho and Grupen's system) as a by-product of learning about activities. This idea takes a little getting used to; much of the following discussion is in support of it.

Figure 3 shows sketches of two concepts. Both begin with one agent, A, running toward another, B. One of these interactions develops into an embrace, the other into a chase. The entailments of these concepts are different, too: the rush-to-embrace concept implies that A and B are happy whereas the frighten-and-chase concept implies that B is unhappy.

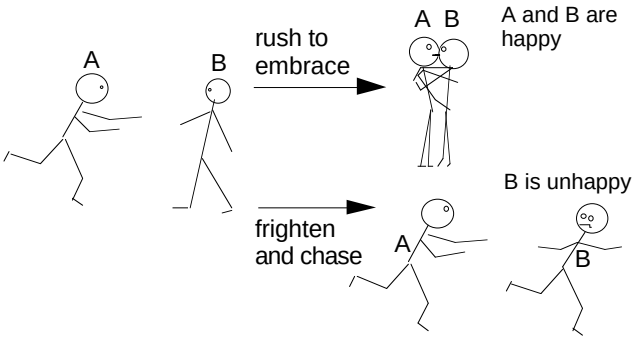

Figure 3. Sketches of oncepts for two activities, rush-to-embrace and frighten-and-chase. The participants or roles are identified (A and B) as are some entailments (e.g., A and B are happy).

Conceptualizations are instantiated concepts, formed by binding the participants in an activity to the *roles* of a concept. Suppose you are at the airport and you see a child running toward an adult getting off the airplane. By binding the child to A and the adult to B in the rush-to-embrace concept, one conceptualizes the activity. Were the

same child to run at another, younger child, the conceptualization and its entailments would be different:

Classes of entities are defined by the roles of concepts. The rush-to-embrace concept, for instance, has at least two roles, one for the entity that rushes to embrace, the other for the entity that receives the embrace. In our experience with the rush-to-embrace concept, we learn extensional classes of entities that give and receive embraces: Humans do, snakes and cars and spoons don't. The child in Figure 4 spends a lot of time reaching, grasping and mouthing objects, so she may form a reach-grasp-mouth prototype, and entailments of the prototype, and also a category of objects that are reached for, grasped, and mouthed.