Robot Learning: A Sampling of Methods

Jonathan Mugan

CS395 Conference Course
May 11, 2005

Overview

On first appearances, one way to enable a robot to learn would be to simply have the robot perform random actions until something interesting happens. The robot could then agglomerate the sequence of actions that lead to the interesting result into a higher-level action and incorporate that higher level action into its set of possible actions. Repeating this process over time this would lead to increasingly complex behavior and eventually to human-level competence. It is of course not that simple, while associational learning is clearly important, the number of challenges that must be overcome to achieve learning of complex behavior seems to grow with each published result.

For a robot to function in a complex world it must learn to do many things that humans take for granted. It must come to understand and use its perceptual system; specifically, it must be able to identify objects and form object categories and to ground those objects and categories in its internal representation (Harnad 1990). It must also learn how to manipulate objects and to anchor (Coradeschi & Saffiotti 2003), or associate, external objects with internal representations of those objects as it performs actions. So the robot does not get lost, it must learn about small and large scale space and navigation. Also, if it is desired that the robot communicate effectively with humans, then it must learn natural language.

These challenges pose specific problems, a few of which will be mentioned here. The sensory input vector is of extremely high dimension and so a lower dimensional substructure consisting of useful features must be found. This lower dimensional substructure must individuate actions and objects so that the robot can reason about them. A compounding problem is that even a very high dimensional sensory input vector will contain only a primitive representation of the environment, and the robot must still learn to function with this impoverished representation. Also, learning how actions affect the sensory features requires identifying which context features are important for enabling the action to have the predicted or desired effect. Additionally, there is the problem of naive induction, the robot must decide which things are worth learning.

This seems like a daunting task indeed. However, some inspiration can be taken from the fact that in the natural world animals with incomplete solutions to many of these problems are able to thrive. For example, most of a frog's perception comes through a very limited vision system that provides its brain with only an exceedingly rudimentary representation of its environment (Lettvin *et al.* 1959). Even humans have in imperfect perceptual system, for example there is blind spot in the area where optic nerve fibers leave the retina (Kandel, Schwartz, & Jessell 2000). More significantly, in (Rensink, O'Regan, & Clark 1997) it was shown that human subjects had difficulty noticing changes in a visual scene when the altered scene was shown after a briefly presented blank scene. This *change blindness* supports the idea that the human brain does not maintain a detailed representation of the environment demonstrating that human-level functioning is possible with a limited representation.

A useful way of looking at the robot learning problem is the equation $\mathsf{behavior} = \mathsf{architecture} \times \mathsf{content}$ (Laird, Lehman, & Rosenbloom 1996). A robot must learn an architecture, content, or both. The methods surveyed in this paper will focus on specifying some form of architecture and then learning content for it. This architecture supplied must be general enough to allow a broad variety of behaviors, but if it is too general then little can be learned. This tradeoff is roughly analogous to the concept of inductive bias as explained in (Mitchell 1997). This survey will focus on particular methods to enable a robot to learn useful concepts and behaviors, and will not be organized by schools of thought. However, it is worth noting that there are two contemporary approaches to robot learning, and they will be discussed in the remainder of this section.

Behavior-Based Robotics

One way of organizing the learning process is to focus on learning specific behaviors. This model can be traced back to the subsumption architecture (Brooks 1986). A recent example is (Ballard *et al.* 1996) in which a system is devised that learns *microbehaviors* in a virtual reality environment. These microbehaviors are state/action tables and are built using reinforcement learning. Multiple microbehaviors may be running at once and so the system also includes methods for microbehavior arbitration and selection. A textbook that emphasizes the behavior-based approach is (Arkin 1998).

Developmental and Epigenetic Learning

Developmental robotics focuses on task independent learning and is inspired and influenced by developmental psy-

chology. Epigenetic robotics is a similar effort except that the emphasis is more on cognitive and social development (Lungarella *et al.* 2004), and here they will be discussed together. An important early work in this area was the schema mechanism (Drescher 1991), and an article in Science magazine (Weng *et al.* 2001) helped to solidify developmental learning as an independent and cohesive area. A recent survey is given in (Lungarella *et al.* 2004). The remainder of this section will highlight some recent work in this area.

Weng (2004), has put forth some broad ideas for developmental robotics. One particular idea is that he argues for the need for a "mind centered" representation as opposed to a "world centered" representation because of the variety of sensory inputs that refer to the same world-centered object. He therefore concludes that a symbolic representation is not suited for developmental learning, and that what is needed is a high-dimensional, mind-centered numeric representation.

In (Gold & Scassellati 2005), a method is described that allows a robot to distinguish when it is looking at itself in its visual field by learning a range for the amount of time that it takes to see the results of its own motion. Thus, if the robot initiates motion and then sees movement within that range of time, then it presumes that it is its own body that it is seeing.

The Constructive Learning Architecture (CLA) (Chaput 2004) is method that uses a hierarchy of self-organizing maps (SOMs) (Kohonen 1995) to model cognitive development. At the lowest level the SOMs are trained on the actual input vectors. As part of this training, a distance is calculated between the input vector and every representative node in the SOM. This distance a_i between input vector x and SOM node i is calculated as

$$a_i = e^{-\frac{|x_i - m_i|^2}{2\sigma^2}}$$

where σ is used as a parameter of how much a_i is affected by distance. When a training vector is presented to a lowest level SOM the vector a can be collected, this vector a is then used as the training vector for the next higher level. This means that for a given SOM that is not at the base level that the ordinality of its nodes must be the sum of the ordinalities of the lower level a vectors that feed into it. The SOMs can be organized into trees and loops with or without time delay. The CLA architecture gives no automatic way for the hierarchy to be formed, the structure must generally be created by hand or by some outside specified algorithm. An important element of CLA is the ability to do fallback. If a particular level of SOM doesn't have a strong activation for a particular input vector then the system just utilizes the lowest level for which there is a sufficient activation.

Learning States and Actions

This section will discuss various methods that allow a robot can learn about the nature of its environment and about how its actions affect the environment. The emphasis for this section will be on the notion of state and how state is learned and represented.

Learning of Deterministic Finite Automata

One way to represent the environment is by using a Deterministic Finite Automata (DFA). The robot can take actions to transition from one state to another, and the output at each state corresponds to what the robot perceives in that state. The problem then is to learn the underlying DFA for an environment. DFA learning can be complicated by various factors: there may be many states that have the same output (ambiguous states), the robot may make errors in sensing the output of the state, and the robot may make mistakes in moving from one state to another.

An algorithm called L^* (Angluin 1987) allows a robot to learn a DFA with ambiguous states if there is a reset that can jump the robot back to a fixed state, and the robot is provided with an oracle that can answer two kinds of questions. The oracle can say if a presented string is in the language, and the oracle can say if the currently constructed DFA is isomorphic to the correct DFA, and provide a counterexample if it is not. To remove the need for a reset, (Rivest & Schapire 1989) use *homing sequences*. A homing sequence is a sequence of actions such that if the outputs observed by executing that sequence of actions from any two states are the same, then executing that sequence of actions from either of those two states leads to the same state.

Learning of DFA is closely related to the problem of causal mapping with a high number of ambiguous states. (Dudek *et al.* 1991) showed that markers were necessary for mapping in a graph-like environment in which the the only things that the robot could sense were if it were at a vertex, the number of outgoing edges of that vertex, and the marker carried by the robot. They gave a polynomial time mapping algorithm using these markers. In (Dudek, Freedman, & Hadjres 1993), they replaced the markers with *extended signatures*. A *signature* is the number of outgoing edges of a vertex, and the extended signature is the number of outgoing edges of the vertices of a node's neighbors, and possibly their neighbors, etc. This procedure is successful for creating a map of the graph except in certain degenerate cases in which outside information must be used.

(Dean *et al.* 1995) put forth an algorithm for learning DFA with ambiguous states and the additional constraint that an incorrect output value for a state may be perceived with probability less than one-half. The algorithm takes advantage of the important assumption, which is that each time the robot visits a state the output perceived may be different, by visiting states multiple times. The algorithm does require a *distinguishing sequence*, which is a sequence of actions such that the sequence of observations created by executing that sequence of actions from any two states will be distinct. All homing sequences are distinguishing sequences but not the other way around, and although all DFA have homing sequences not all DFA have distinguishing sequences.

From DFAs to POMDPs and PSRs

As described in (Cassandra 2004), a partially observable Markov decision process (POMDP) is a Markov decision process (MDP) in which the robot is unsure of its current state. It uses the sets of states, actions, transitions, and re-

wards of the MDP, but adds a set of observations. The observation received at a state gives the robot a clue about what state it is in. This is done through the observation model, which gives the probability of seeing each observation in each state. At each time t_i the robot maintains a belief state, which is a probability distribution over all states reflecting where the robot thinks it might be. This process of moving between belief states via actions can be modeled as an MDP over the belief states since the next belief state only depends on the current belief state, the action, and observation (although this is complicated by the fact that the value function is a continuous function over the belief space).

Rivest and Schapire (1994) investigated a diversity-based representation of finite automata. The *diversity* of a DFA is the number of test equivalence classes, meaning the number of sets of tests that always give the same value. They point out that for many real world problems with structure, that the set of test equivalence classes for a DFA is much smaller than the number of states. These ideas have been transferred to the predictive state representation (PSR) framework. As described in (Littman, Sutton, & Singh 2001), a PSR is defined as a set of *tests* consisting of sequences of actions and observations that provide sufficient information for a robot to know the results of all other possible tests. The state of the system is represented as a vector of probabilities of seeing the predicted observations given that the actions of the tests were performed. In (Littman, Sutton, & Singh 2001), it was shown that any environment that can be represented with a POMDP can also be represented with a PSR, and further that the number of the tests of the PSR would not be larger than the number of states of the POMDP.

A PSR can be thought of as representing the state of the world not by a node in a graph or by what the robot has seen in its history, but rather would would happen if the robot performed various action sequences; or, more specifically, the probability of various actions sequences having their stated effect. In (Singh *et al.* 2003) a gradient-based algorithm is presented for learning to make accurate predictions of tests, however they state that currently no algorithm exists for determining a sufficient set of tests. An advantage of PSRs is that their predictions are observations and are thus directly observable, but PSRs have the disadvantage of being an unintuitive representation of a robot's state.

Hierarchical Reinforcement Learning

Traditional reinforcement learning methods generally only do well in situations in which there are relatively few states (low dimensionality) and in situations in which the tasks to be learned require relatively few actions (low diameter). Hierarchical reinforcement learning is one proposed method to alleviate these difficulties, and the excellent survey is (Barto & Mahadevan 2003).

In (Sutton, Precup, & Singh 1999) hierarchy is achieved through options. An *option* is similar to a subroutine and consists of three parts. An *option policy* that is a policy for behavior that is used while the option is running, an *initiation set* consisting of the states from which the option can be initiated, and the *termination condition* that specifies when the option terminates. Also there are *option models* which give a probabilistic description in what state the option will terminate and the total reward for running the option. Additionally, there are *intra-option learning methods* that allow the policies of many options to be updated simultaneously. Options are generally pre-defined, however in (Singh, Barto, & Chentanez 2005; Stout, Konidaris, & Barto 2005), instead of the options begin pre-specified, a salient event triggers the creation of an option and its option model. The initiation set initially consists of the state leading to the state that created the salient event. As more states are found to chain to that set of states the initiation state set is expanded to include those states. This is very similar to Drescher's notion of *composite actions* as will be discussed later in the paper.

Other hierarchical reinforcement learning methods include Hierarchies of Abstract Machines (HAMs) (Parr 1998; Parr & Russell 1998), and MAXQ Value Function Decomposition (Dietterich 2000).

Bootstrap Learning

The Spatial Semantic Hierarchy (SSH) (Kuipers 2000) is a method that a robot can use to build a topological map of its environment and to reason about space. The SSH consists of multiple levels each with its own ontology. The lowest level is called the *control level* and consists of trajectory following and hill climbing control laws that take the robot from one distinctive state to the next. Starting from a state the robot trajectory follows to the neighborhood of the next state and then hill climbs to reach that next state. The goal of this strand of bootstrap learning is to enable a robot that begins with no knowledge of its sensors and effectors to learn enough about itself and its environment to be able to reach the first (lowest) rung of the SSH.

Uninterpreted Sensors and Effectors In (Pierce & Kuipers 1997) a simulated robot that was initially endowed with no knowledge of its sensors and effectors was able to learn to use those sensors and effectors, and to eventually be able to move between distinctive states. The first thing that the robot needed to learn was what sensory modalities it possessed. It did this by using agglomerative clustering with a custom distance metric and a termination criterion on all the time series data streaming from its input vector. This time series clustering grouped items belonging to the same modality together.

Once the sensory modalities were partitioned, the robot performed a sequence of steps to determine if any of the modalities that was composed of multiple sensory elements had a shape. It found one such modality, the distance sensors, with multiple sensory elements and went about determining if it had a shape. It did this by creating a distance matrix composed of the distances between the time series corresponding to sensory elements in that modality. It used a distance metric which consisted of the L_1 norm divided by the number of observations. The robot then performed metric scaling on that matrix using a scree diagram to determine the best number of dimensions. It then ran a relaxation algorithm to refine the shape, and what emerged from the distance sensor modality was a replication of the circle representing the physical locations of the sensors on the robot.

This was a critical point because it was where space entered the ontology. Before, all that existed was time and temporal derivatives, but now, using the shape of the distance sensors, spatial derivatives could be computed.

Using these spatial derivatives, the robot was able to learn a model of its motor apparatus. It did this by first partitioning the infinite space of all motor control vectors into a finite set of representative motor control vectors that were uniformly distributed through the space. It then randomly executed these motor control vectors for a period of time keeping track of the average motion vector field for each movement. The robot then used principal component analysis to analyze the space of these average motion vector fields and came up with a set of principal eigenvectors that captured the effects of the motor apparatus. The robot then found the representative motor vectors that matched the principal eigenvectors. Matches were found for the top two eigenvectors and corresponded to turning in place and moving forward.

Once the robot had a sensor and an effector model it could define features and local state variables. Local state variables are features that can be described as a function of the motor control vector. The robot found that the local minimum of the distance sensor could be described with such a function. The robot found that if the local minimum was located in a distance sensor in front of the robot then when the robot went forward its value decreased, and that the opposite occurred if the local minimum was located behind the robot. The robot also found that if the local minimum was on one of its sides, then that local minimum value often remained constant as the robot went forward.

In order to define open-loop path-following behaviors based on this local minimum, the robot used an arbitrary target value for the distance of the local minimum. This enabled the robot to hillclimb to this value. It could also trajectory follow by keeping a local minimum as close to this value as possible. In order to define closed-loop path following behaviors, the robot performed the open-loop path-following behaviors with various other actions mixed in the see how the other actions affected the local minimum. From this, it learned how to adjust the value of local minimum if is strayed too far from the target value. Using these closed-loop path following behaviors, the robot then had bootstrapped itself to the bottom rung of the spatial semantic hierarchy (SSH) (Kuipers 2000).

Finding Distinctive States with a Growing Neural Gas Once a robot has a sensor and effector model, the method outlined in (Provost, Kuipers, & Miikkulainen 2004) can be used to remove the need for built in control laws. The method they constructed used a growing neural gas SOM (Fritzke 1995) to partition the input space into distinctive states. A limited number of actions were then defined to move between these states. The actions that they used were *forward*, *backward*, *turn-left*, and *turn-right* and each consisted of trajectory following and then hill climbing. Trajectory following consisted of traveling or turning until a different node of the SOM became the closest node to the sensory vector, and hill climbing consisted of following the gradient such that the sensory input vector comes as close as possible to the value of the new current node. A standard reinforcement learning algorithm was then run using those distinctive states and actions. The hill climbing is an area in which they are working to improve, currently the robot must take a step in each direction to find the gradient.

This approach can also be used to reduce diameter and the dimensionality of a problem so that reinforcement learning may be used more effectively. The dimensionality of the problem is reduced by using the growing neural gas nodes as states and the diameter is reduced by using the limited number of actions to move between those states.

Place Recognition (Kuipers & Beeson 2002) improved the robustness of place recognition using bootstrap learning. One challenge in place recognition is *image variability*, the problem of the same place never looking the same due to the high dimensionality of the sensors; and another challenge is *perceptual aliasing*, the situation in which many different places look the same. Their approach to dealing image variability was to first cluster the sensory values at distinctive states into a fixed number of clusters. Since their method for choosing the number of clusters generally gave a number that was fewer than the number of distinctive states, perceptual aliasing was increased. To overcome the problem of perceptual aliasing, they then performed topological mapping on this space. This removed the perceptual aliasing since topological mapping uses the relative positions of the states with respect to each other to disambiguate them. The output from the topological mapping was then used as a supervisory signal for the supervised learning of the association between perceptual views to states. This then allowed the robot to know where it was without having to resort to exploration to disambiguate its state.

The Schema Mechanism

The schema mechanism (Drescher 1991) is a method by which a robot can learn a representation of how actions affect the world. The system is made up of three core elements. An *item* is a predicate that can be either *On* or *Off*. The values of the items tell the robot everything it knows about the state of the world. An *action* is something that the robot can do to affect the world. Finally, a *schema* is a triple composed of a context, action, and result. For a schema, both the context and the result are composed of items that can either be *On* or *Off*. A schema says that if the context items are satisfied and the action is taken, then the result item will take on its specified value with some probability. The goal of the system is to develop schemas that are reliable. Using the example in (Chaput 2004), the schema

$$\langle \mathsf{InFrontOfDoor} | \mathsf{OpenDoor} | \mathsf{DoorOpen} \rangle$$

would mean that if the robot were in front of the door, and it opened the door, then the door would be open.

Initially, the robot starts off with a set of *primitive items* corresponding to both coarse sensory input and proprioperception that are updated automatically by the system. The robot also begins with a set of *primitive actions* that corresponds to simple movements, and each such primitive action also serves as the action for a schema with an empty context and an empty result called a *bare schema*.

To begin the learning process the robot executes its primitive actions and by keeping track of correlations between all actions and all items, the system incrementally creates new schemas that bring about results with increasing reliability.

In order to add to the ontology, when a schema is found to be unreliable but locally consistent (meaning that when its schema is successful in bringing about its results it will continue to be successful for a period of time afterwards) the schema itself is reified as a *synthetic item* and can then become part of the context or result of a schema just like the primitive items. Additionally, when a schema is created that has an item in its result that does not exist in any other schema then a *composite action* is created. A composite action is like a subroutine, it allows the robot to perform high-level actions. When a composite action is created a *controller* is created for it that backchains to find applicable schemas that lead toward the goal. When a schema containing the composite action is is chosen for execution, the *controller* continuously activates the applicable schema closest to the goal until the goal is achieved.

A composite action is quite similar to an *option* in reinforcement learning (Sutton, Precup, & Singh 1999), both serve as closed-loop control laws to achieve some high-level goal. Composite actions are created each time a novel result is obtained, this is similar to how options are created in (Singh, Barto, & Chentanez 2005; Stout, Konidaris, & Barto 2005), except there the results that trigger the creation of an option are predefined "salient" events.

When the schema mechanism was run on the simple *microworld* (a 7×7 grid consisting of an immobile agent, a hand, an eye, and two items) the robot learned a model for how the location of objects in the visual field would move as the glance moved, as well as models for the movement of its hand and glance. It was also able to learn such things as how to bring its hand to its mouth and how to shift its gaze to bring objects to the fovea.

The schema mechanism does not represent what the robot should do in a given situation, but rather what would happen if it performed some action. It builds schemas by by finding results that are only slightly more likely to occur after a specific action then otherwise, it then hillclimbs by incrementally finding context conditions that make that result occur more reliably. The difficulty however, is its computational complexity. To store all of the correlations it requires space equivalent to the number of schemas times the number of items, and after each action it requires time equivalent to a subset of the schemas times a subset of the items. These subsets are not necessary small. A potential method for alleviating the computational complexity problem of the schema mechanism is CLASM, which is discussed next.

The Cognitive Learning Architecture Schema Mechanism (CLASM) CLASM replaces the process of marginal attribution by allocating a SOM for each action. The vectors of these action SOMs consist of the extended context and extended result of the action at the time the SOM was created. After a suitable training period, nodes of the action SOM that contain result items above a certain threshold are harvested. Each harvested schema then becomes reified as a synthetic item and a composite action is created for each new result as part of some harvested schema. The resources of the original SOM level are then released and the training of the next level begins using all the original actions and the new composite actions. Additionally, each node then contains weights for all the original items plus the new synthetic items. When CLASM was applied to the microworld of Drescher it was able to learn the same things as in Drescher's implementation.

CLASM is more efficient because it eliminates the trail of context schemas, a schema with three context items can be added all at once. Additionally, it eliminates many of the ad hoc methods Drescher proposes to mitigate exponential growth of schemas. For example, there is no need to keep track of whether one schema is more specific than another, and multiple results can be added without needing to be in the context of another schema. However, there is one important difference in the behavior of the generation of new schemas. In CLASM, if a particular item is always *On* when an action is taken then the schemas created using that action will contain that item, however in Drescher's implementation they will not because what is taken into account is the ratio of success when *On* compared with *Off* and not just the correlation.

Focus of Attention and Motivation

A typical robot learning system will instruct the robot to execute various actions in various situations to learn the effects of those actions in those situations. Ideally, one would want the robot to explore the areas of the state and action space that could teach it the most. One guiding principle for formulating this is to motivate the robot to learn in areas in which it is least able to predict the outcomes of its actions (Schmidhuber 2005). However, some areas may be so difficult that no learning can take place. This leads to the formulation of a second principle that the robot should be motivated to explore areas of its state and action space based on its ability to improve its predictions in those spaces (Schmidhuber 2005).

An example of the first principle is (Singh, Barto, & Chentanez 2005; Stout, Konidaris, & Barto 2005). In that body of work the robot is intrinsically rewarded for bringing about unexpected salient events, with the reward being directly proportional to the degree that the event was unexpected. This high reward will cause the robot to to bring about this state often, but as it does is unexpectedness will diminish and it will go on to new behaviors. This is very similar to the concepts *habituation* and *hysteresis* in (Drescher 1991). When activating schemas habituation causes schemas that have been activated many times in the past to be activated serving as a focus of attention, but hysteresis allows the robot to go onto new behaviors after the current behaviors are sufficiently learned.

An example of the second principle is (Oudeyer *et al.* 2005). They approach the problem of endowing intrinsic motivation using what they refer to as Intelligent Adaptive Curiosity (IAC). In this approach, the memory of all experiences is split into regions based on exemplar vectors of experience. Each region has a learning machine called an

expert, the expert is trained on the exemplars in a region. When a prediction corresponding to a particular region has to be made by the robot it calls on the expert of that region to make the prediction. The prediction is compared with the actual result and that difference is noted in a list. This list is then used to approximate the derivative of the the error curve in that area. Actions are then chosen based on where the derivative is the steepest.

Learning About Objects and their Meanings

Methods such as Shanahan's abductive account of perception (Shanahan 2005) exist for enabling the robot to reason once the world has been broken up into objects and relations, but no comprehensive method exists for finding such discretizations. One promising direction of research, however, is active vision. (O'Regan & Noë 2001) argued in their landmark work that instead of being used as a passive receptor, vision is used to actively explore the world and that humans use sensorimotor contingencies of the visual system to learn about the objects in their environment. This embodied learning approach leads to one method for symbol grounding by allowing symbol meaning to be encoded in visual stimuli.

Grounding Words in Visual Representations

(Roy & Pentland 2002) in their CELL model used the fusion of a visual and an auditory sensory stream to learn an association between the verbal and the visual representation of objects, effectively grounding the verbal representation to the visual one. Using an input stream of natural caregiver speech that occurred while a caregiver played with an infant, and a visual stream of the object with which they were playing, CELL was able to associate the phonetic representation of the words with the objects. The system did both speech segmentation and word matching, and associated words with objects by exploiting the fact the the word for the object would often be uttered while the object was in the visual scene.

The work of (Yu & Ballard 2004a) was similar in the respect of associating temporally co-occurring objects and words, however, word segmentation was done separately and each scene contained multiple objects. This temporal co-occurrence was also exploited in (Yu & Ballard 2004b) with the addition that objects and actions were grounded with the aid of deictic references. These deictic references (Ballard *et al.* 1996) help to reduce the space of possibilities for word association and allow more effective learning of words (Yu, Ballard, & Aslin 2003).

(Steels 1998) in his talking heads experiment created a system in which two robots watching a scene create a lexicon to describe the objects within. Each robot identified image segments (objects) in the scene and created a feature vector for each object. To add words to the lexicon, one robot choose one object that had a subset of distinctive features as the topic and sent a word, represented as letters, to the other robot. The other robot checked to see which subsets of distinctive features were in its view and associated that word with each such subset. Based on the success of these associations a shared lexicon was created.

Manipulating Objects and Affordances

An *affordance* is described in (Gibson 1979) as what the environment offers for manipulation given the physical characteristics of the observer. The affordances of the environment exist independent of the observers ability to perceive them, and they do not change as the goals of the observer change.

In (Modayil & Kuipers 2004) an algorithm is put forth that allows a robot to track and categorize movable objects using a range sensor. They use an occupancy grid, and if a particular square has ever been empty then any subsequent reading falling into that square is considered to be part of a movable object. Such readings that are near to one another are merged into possible objects that are tracked as they move or the robot moves, or both. The shapes of objects are then determined by sensing the objects at different angles using a range sensor. The objects are then clustered by shape using online clustering.

(Stoytchev 2005) has a system in which a robot has a set of exploration behaviors relative to objects. Each body part has an associated Cartesian coordinate system. When a behavior is initiated with a body part, a set of invariant functions is run to see if the movement of the object corresponds to any of the invariants, meaning that the object was moved by the body part.

Modeling Sensory Input: Neural Models of Vision

Vision is becoming increasingly important for robotics, but making using of visual information is an extremely difficult problem. However, if we were able to successfully model human vision, than such a model might give us a handle on the problem of creating useful robot vision systems. This section describes two such modeling efforts.

The HLISSOM model (Bednar & Miikkulainen 2004) is a hierarchy of sheets of neural nets that models the early visual pathway. The model contains a sheet of photoreceptors, the LGN OFF and ON sheets, and the V1 area. They performed an experiment in which they simulated both the prenatal and postnatal stages of development. In the prenatal stage they presented noisy patters of neural activity containing a disk, and in the postnatal stage they presented natural images. Significantly, they found that the postnatal map was a refinement of the prenatal one without changing the overall shape. Also worth noting is that the postnatal map was biased towards horizontal and vertical orientations.

Olshausen (2003), argues that the visual cortex uses a probabilistic model of images, and he puts forth a model that uses a linear superposition of features. He states that redundancy reduction has been an reasonably good framework to describe the way that the neurons in the retina and the LGN respond to stimulus. In traveling from the retina and the LGN to the visual cortex the information from the 100 million photoreceptors must pass through the optic nerve which contains approximately one million ganglion cells. Thus, the redundancy reduction model seems to fit well with the need to optimize the limited resource of the optic nerve. However, the V1 area of the visual cortex has many more outputs than there are inputs in the optic nerve, and he concludes that re-

dundancy is being increased in the cortex. He argues that a *sparse code* could be used to model V1, meaning that only a small fraction of the neurons would be active at any moment. The neurons would form basis and any visual image could be represented as a small subset of all the basis. The model is of the form

$$I(x) = \sum_i a_i \phi_i(x) + \nu(x)$$

where $I(x)$ is an image patch, ϕ_i is a basis function, a_i is the coefficient, and $\nu(x)$ is the Gaussian noise. The image patch is represented as the vector a and it is important to note that this basis set is *over-complete*, meaning that there are more a_i than there are elements in the image patch. This means that there are an infinite number of ways to describe an image. For a particular image the a vector is chosen that maximizes $P(a|I,\theta)$ where θ is a vector of noise parameters. By Bayes rule we have

$$P(a|I,\theta) \propto P(I|a,\theta)P(a|\theta)$$

and the MAP estimate of $\hat{a}$ can be computed by gradient descent.

He carried out a method to compute the basis functions on hundreds of thousands of image patches from natural scenes and got a set of spatially localized, oriented, bandpass functions. He also extended this model to incorporate time by using sequences of images with a format given by

$$I(x,t) = \sum_i \sum_t{}' a_i(t')\phi_i(x, t-t') + \nu(x,t)$$

and he was again able to get a set of spatially localized, oriented, bandpass functions, however in this case they translated over time.

Transfer Learning

Robot learning requires more flexibility than traditional, one-task machine learning programs provide. The book *Learning to Learn* (Thrun & Pratt 1998), covers many examples of learning algorithms whose performance on a particular task improves both with experience on that task and experience on related tasks. The idea is that having access to related tasks in the domain can allow the algorithm to learn invariants that can help with all tasks. Most of the methods in the book are based on neural networks, and within that framework there are two basic approaches. *Representational transfer* refers to methods that use the network trained on a previous task as the starting point for learning the new task, and *functional transfer* refers to task transfer that is done on multiple tasks simultaneously.

In (Caruana 1997), learning multiple tasks at once was shown to give higher overall performance than learning a single task at a time. In the most basic implementation, multitask learning (MTL), the multiple tasks were trained in parallel on a neural network using a shared hidden layer. This performed better than single task learning (STL) because learning multiple tasks in parallel allows internal representations in the hidden layer that arise while learning one task to be used for other tasks. He showed that it was indeed this multitask effect that caused the increased performance, and not some other factor, by looking at the performance of a single task when all of the other tasks had randomly reassigned training signals. He found that the shuffling brought the performance of MTL down to be comparable with STL. He also ruled out that MTL performs better due to restricted net capacity by running STL at various net sizes.

Memory-based Methods

In memory-based methods the robot stores all of its experience and then decides on an action for a given state and desired outcome based on an interpolation of similar state-action-outcome tuples in memory. The mathematics for this method is explained in (Atkeson, Moore, & Schaal 1997a), and in (Atkeson, Moore, & Schaal 1997b) the methods are applied to various problems such as billiards and devil sticking. The simplest case is for temporally independent tasks. Given a state x, and an action u, the output y can be represented by the equation

$$y = f(x,u) + \mathsf{noise}.$$

Thus an inverse model in which a state x and a desired output y gives the necessary action u is written as

$$u = \hat{f}^{-1}(x,y).$$

If the (x_i, y_i, u_i) tuples are listed in a database, then the correct u_i can be found by finding the tuple(s) with the closest x_i, y_i values. A disadvantage of the inverse model is that it can go awry in non-monotonic environments. Therefore, a forward model $y = \hat{f}(x,u)$ can also be used, however it is more complicated to derive the action u_i based on the output y_i and the state x_i, because a numerical inversion of the forward model is required. The inverse model can provide a seed for this inversion, and this is the process that was used for the task of billiards. The system learned to sink another ball with the white ball with roughly 75% accuracy in 100 trials.

The other application of devil sticking was more complicated because the tasks were temporally dependent. Deadbeat control (control that attempts to fix the error in one action) was found to be insufficient. They implemented a *linear quadratic regulation* (LQR) controller and found that it worked for devil sticking with manual generation of training data to estimate the matrices of the local linear model but required a manual search for an equilibrium point. These problems were alleviated by using the *shifting setpoint algorithm* (SSA).

Search Trees for Feature Discovery

One important open problem is how to devise a systematic way for a robot to discover useful features in its sensory stream. One potential approach to this problem is for the robot to perform a search through the space of features using a pre-defined set of statistical methods as operators and applying those operators to current features to create new features.

The robot would begin with its set of type-specific statistical methods and the raw sensor stream as the first feature.

It would then do a breadth first search through the space of possible features using the statistical methods as operators to create new features. These operators being strongly typed would help to limit the search space. In deriving a new feature, an operator could use the data from any of the predecessors of the current feature.

For example, in (Pierce & Kuipers 1997) this method was used to find the local minimum distance sensor as a useful feature. That branch of the search tree proceeded as follows. A group generator (clustering) was applied to the raw sensor feature to partition the input vector into sensory modalities. Then, an image feature generator was applied and it found that one group of sensors formed a circle. The local minimum operator was then applied, and once the local minima were identified the tracker operator was applied to identify the location of the local minima relative to the robot. With this feature the robot was able to learn homing and path following behaviors.

This is one viable approach to generate the features, but what is needed is a robust and efficient way to determine the usefulness of a feature. There are really two issues here, one is the coherence of the feature, does the feature make any sense? Searches that go through incoherent features can be abandoned. The second is the usefulness of a feature. A confounding factor is that sometimes the feature itself may not be directly useful to the robot, but features that can be derived from it may be. Pierce and Kuipers (1997) used the idea that changes in the feature value should be predictable based on the output vector as a metric for usefulness, coinciding with their goal of navigation.

Statistical Learning Methods

Statistical methods appear to be becoming increasingly important in robot learning. Bayesian methods have been particularly important. Using Bayesian methods, robots can make predictions by averaging the predictions of all the possible hypothesis weighted by the probability of each hypothesis being correct. However, this can be cumbersome, so what is often used is the maximum a posteriori (MAP) hypothesis. This means using only the most likely hypothesis is used to make a prediction. Another simplification is to assume that all of the hypothesis initially have the same probability of being true. Under this assumption, in order to choose the MAP hypothesis it is only necessary to choose the hypothesis h_i that has the maximum value for $P(data|h_i)$. This hypothesis is called the maximum-likelihood (ML) hypothesis and is a quick and simple way for a robot to make decisions in an uncertain environment.

The expectation maximization (EM) algorithm is also very commonly used in robot learning (Thrun 2002). Various other missing variables can be added to the algorithm. For instance, in (Law, Jain, & Figueiredo 2003) missing variables are added to do feature selection.

Regression Methods

Statistical regression is a classic technique for function approximation. It is also important memory-based robot learning methods as was discussed previously. *Logistic regression* calculates the probability of an item being in class A divided by the probability of an item being in class B, which produces values between $(-\infty, \infty)$. *Kernel regression* finds constant values using locally weighted training examples (Atkeson, Moore, & Schaal 1997a). It gives the estimated value $\hat{y}(q)$ by finding the weighted average at a point q using a kernel function K and a distance function d

$$\hat{y}(q) = \frac{\sum y_i K(d(x_i, q))}{\sum K(d(x_i, q))}$$

Locally weighted regression done as in global regression except that the points are weighted by a kernel function relevant to their distance from some reference point q. It tends to work better than kernel regression near the edges. *Ridge regression* is used if there are not enough equations to solve for the unknown parameters in locally weighted regression. To keep the matrix of points from becoming singular, it adds a matrix with small positive values along the diagonal.

Manifold Learning

Sensory input data is generally of very high dimension and it is therefore useful and often necessary to reduce its dimensionality. If the data exist in some lower-dimension manifold there are many ways of doing this. One classic linear method is *principal components analysis* (PCA), which finds the orthogonal directions of maximum variance. *Multidimensional scaling* (MDS) is another classical method. It finds a lower dimension approximation that preserves the mutual closeness that exists in some higher dimension space. One unique characteristic is that the process does not require the data points themselves, only the distances between them. *Locally linear embedding* (LLE) (Saul & Roweis 2003) is a nonlinear method that assumes that the data are sampled from a manifold. Each data point X_i is approximated by linear reconstruction using the weights W_{ij} of its k neighbors, or those within an ϵ (Euclidean) in the original data space. Then for each data point X_i, the same weights W_{ij} are then used to identify the lower dimensional representation Y_i that has the lowest error as given by:

$$\Phi(Y) = \sum_i \left(Y_i - \sum_j W_{ij} Y_j \right)^2$$

Another nonlinear method is *Isomap* (Tenenbaum, de Silva, & Langford 2000), which uses the concept of geodesic distance. This use of geodesic distance allows it to represents manifolds such as the "Swiss roll." It first connects each point with either its k nearest neighbors or those within some ϵ in Euclidean input space. Then it calculates the shortest path distance between all pairs of points using these connections. It then performs MDS using the shortest path as a measure of distance between two points. *Independent component analysis* (ICA) seeks to find independent sources of information in the data. As described in (Lee *et al.* 1998), if $s(t)$ is a sense vector received with each scalar value consisting of a linear combination of m sources as expressed by the $N \times M$ matrix A, then

$$x(t) = As(t)$$

and the goal is to find a linear transformation W such that the independent components are found

$$u(t) = Wx(t) = WAs(t)$$

This is solved by various methods that maximize entropy to minimize the mutual information between sources.

Additional Methods

Mixture models consist of mixtures of different probability density functions, usually Gaussians, in different proportions. If the covariance matrices are constrained to be a circular then it has the form of a radial basis expansion. To obtain a *radial basis function*, a small number of centroids is first obtained via k-means clustering or some other method. Then the data is recoded with the centroids as a basis. If there are k centroids, weights w, kernel functions K, and a distance metric d, then the estimated value of x, denoted by $\hat{f}(x)$, is given by

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x))$$

Given the basis functions the weights can be learned easily.

Dynamic time warping assumes a set of template time series. When a new time series needs to be classified it is compared with the template time series. For each comparison between time series i and template j they are compared at every possible time combination at a cost of $O(|t|^2)$. This allows time series that our out of phase but similar otherwise to be classified as the same.

Another useful statistical technique to learn a distance metric. As described in (Xing *et al.* 2002), if one is given certain points in the input space that are considered to be similar, it is possible to learn a distance metric based on how those points are similar. If given the problem of learning a distance metric of points x and y of the form

$$d(x, y) = d_A(x, y) = \left\|x - y\right\|_A = \sqrt{(x-y)^T A (x-y)}$$

the optimization problem is to learn a matrix A that best respects the distances given. Note that if A is the identity matrix then this is equivalent to the standard Euclidean distance.

Finally, data compression can be achieved with *vector quantization*, which associates each input with a codeword to represent that input. Is a form of lossy compression. (Linåker & Niklasson 2000) put forth the resource-allocating vector quantizer (RAVQ) that is designed to segment a sequence of vector inputs and to represent each sequence type with a vector model. Specifically, it segments sequences with stable periods and abrupt transitions. It does this by maintaining a moving window of the n most recent input vectors. It adds the moving average of this window as a new vector model when the distance between the moving average and the closest vector model is greater than a threshold and when the current moving average is considered to be stable.

Conclusion

Many pieces of the puzzle have been represented in this sampling of robot learning methods. At times it appears that all that is needed is to find the right way to put it all together, and at other times it seems as if something very profound is missing. Time will tell which conjecture is correct.

References

Angluin, D. 1987. Learning regular sets from queries and counterexamples. *Inf. Comput.* 75(2):87–106.

Arkin, R. C. 1998. *An Behavior-based Robotics*. Cambridge, MA, USA: MIT Press.

Atkeson, C. G.; Moore, A. W.; and Schaal, S. 1997a. Locally weighted learning. *Artificial Intelligence Review* 11(1/5):11–73.

Atkeson, C. G.; Moore, A. W.; and Schaal, S. 1997b. Locally weighted learning for control. *Artificial Intelligence Review* 11(1/5):75–113.

Ballard, D. H.; Hayhoe, M. M.; Pook, P. K.; and Rao, R. P. 1996. Deictic codes for the embodiment of cognition. *Behavioural and Brain Science*.

Barto, A., and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Discrete event systems* 13(4):341–379.

Bednar, J. A., and Miikkulainen, R. 2004. Prenatal and postnatal development of laterally connected orientation maps. *Neurocomputing* 985–992.

Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Trans. on Robotics and Automation* RA-2(1):14–23.

Caruana, R. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

Cassandra, A. 2004. Tony's pomdp page. http://www.cassandra.org/pomdp/tutorial/index.shtml.

Chaput, H. 2004. *The Constructivist Learning Architecture: A Model of Cognitive Development for Robust Autonomous Robots*. Ph.D. Dissertation, University of Texas at Austin, Department of Computer Sciences. Also available as UT AI TR04-34.

Coradeschi, S., and Saffiotti, A. 2003. An introduction to the anchoring problem. *Robotics and Autonomous Systems* 43(2-3):85–96.

Dean, T.; Angluin, D.; Basye, K.; Engelson, S.; Kaelbling, L.; Kokkevis, E.; and Maron, O. 1995. Inferring finite automata with stochastic output functions and an application to map learning. *Machine Learning* 18(1):81–108.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research* 13:227–303.

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

Dudek, G.; Jenkin, M.; Milios, E.; and Wilkes, D. 1991. Robotic exploration as graph construction. *IEEE Trans. on Robotics and Automation* 7(6):859–865.

Dudek, G.; Freedman, P.; and Hadjres, S. 1993. Using local information in a non-local way for mapping graph-like worlds. In *Proc. 13th Int. Joint Conf. on Artificial Intelligence (IJCAI-93)*, 1639–1645.

Fritzke, B. 1995. A growing neural gas network learns topologies. In Tesauro, G.; Touretzky, D. S.; and Leen, T. K., eds., *Advances in Neural Information Processing Systems 7*, 625–632. MIT Press.

Gibson, J. J. 1979. *The Ecological Approach to Visual Perception*. New Jersey, USA: Lawrence Erlbaum Associates.

Gold, K., and Scassellati, B. 2005. Learning about the self and others through contingency. In *AAAI Symposium on Developmental Robotics*.

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

Kandel, E. R.; Schwartz, J.; and Jessell, T. M. 2000. *Principles of Neural Science*. McGraw-Hill Companies.

Kohonen, T. 1995. *Self-Organizing Maps*. Berlin; New York: Springer.

Kuipers, B., and Beeson, P. 2002. Bootstrap learning for place recognition. In *Proc. 18th National Conf. on Artificial Intelligence (AAAI-2002)*, 174–180. AAAI/MIT Press.

Kuipers, B. 2000. The spatial semantic hierarchy. *Artificial Intelligence* 119(1-2):191–233.

Laird, J.; Lehman, J. F.; and Rosenbloom, P. 1996. *A Gentle Introduction to Soar, an Architecture for Human Cognition*.

Law, M. H.; Jain, A. K.; and Figueiredo, M. A. T. 2003. Feature selection in mixture-based clustering. In S. Becker, S. T., and Obermayer, K., eds., *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press. 625–632.

Lee, T.-W.; Girolami, M.; Bell, A. J.; and Sejnowski, T. J. 1998. A unifying information-theoretic framework for independent component analysis.

Lettvin, J.; Maturana, H.; McCulloch, W.; and Pitts, W. 1959. What the frog's eye tells the frog's brain. *Proc. Inst. Radio Engrs. NY* 47:1940–1951.

Linåker, F., and Niklasson, L. 2000. Sensory flow segmentation using a resource allocating vector quantizer. In *SSPR/SPR*, 853–862.

Littman, M. L.; Sutton, R. S.; and Singh, S. P. 2001. Predictive representations of state. In *NIPS*, 1555–1561.

Lungarella, M.; Metta, G.; Pfeifer, R.; and Sandini, G. 2004. Developmental robotics: a survey. *Connection Science* 0(0):1–40.

Mitchell, T. M. 1997. *Machine Learning*. New York: McGraw-Hill.

Modayil, J., and Kuipers, B. 2004. Bootstrap learning for object discovery. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

Olshausen, B. A. 2003. Principles of image representation in visual cortex. In L.M. Chalupa, J. W., ed., *The Visual Neurosciences*. MIT Press. 1603–1615.

O'Regan, J. K., and Noë, A. 2001. A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences* 24(5).

Oudeyer, P.-Y.; Kapalan, F.; Hafner, V.; and Whyte, A. 2005. The playground environment: Task-independent development of a curious robot. In *AAAI Symposium on Developmental Robotics*.

Parr, R., and Russell, S. 1998. Reinforcement learning with hierarchies of machines. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, 1043–1049. Cambridge, MA, USA: MIT Press.

Parr, R. 1998. Hierarchical control and learning for markov decision processes.

Pierce, D. M., and Kuipers, B. J. 1997. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence* 92:169–227.

Provost, J.; Kuipers, B. J.; and Miikkulainen, R. 2004. Self-organizing distinctive-state abstraction learns perceptual features and actions. In *AAAI-04 Workshop on Learning and Planning in Markov Processes*.

Rensink, R. A.; O'Regan, J. K.; and Clark, J. J. 1997. To see or not to see: The need for attention to perceive changes in scenes. *Psychol. Sci.* 8:368–373.

Rivest, R. L., and Schapire, R. E. 1989. Inference of finite automata using homing sequences. In *Proc. 21st ACM Symposium on Theory of Computing*, 411–420. ACM.

Rivest, R. L., and Schapire, R. E. 1994. Diversity-based inference of finite automata. *Journal of the ACM* 41(3):555–589.

Roy, D. K., and Pentland, A. P. 2002. Learning words from sights and sounds: a computational model. *Cognitive Science* 26:113–146.

Saul, L. K., and Roweis, S. T. 2003. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *J. Mach. Learn. Res.* 4:119–155.

Schmidhuber, J. 2005. Self-motivated development through rewards for predictor errors/improvements. In *AAAI Symposium on Developmental Robotics*.

Shanahan, M. 2005. Perception as abduction: turning sensor data into meaningful representation. *Cognitive Science*. to appear.

Singh, S.; Barto, A. G.; and Chentanez, N. 2005. Intrinsically motivated reinforcement learning. In Saul, L. K.; Weiss, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press.

Singh, S.; Littman, M. L.; Jong, N. K.; Pardoe, D.; and Stone, P. 2003. Learning predictive state representations. In *Proceedings of the Twentieth International Conference on Machine Learning*.

Steels, L. 1998. The origins of syntax in visually grounded robotic agents. *Artificial Intelligence* 103:133–156.

Stout, A.; Konidaris, G.; and Barto, A. 2005. Intrinsically motivated reinforcement learning: A promising framework for developmental robot learning. In *AAAI Symposium on Developmental Robotics*.

Stoytchev, A. 2005. Toward learning the binding affordances of objects: A behavior-grounded approach. In *AAAI Symposium on Developmental Robotics*.

Sutton, R. S.; Precup, D.; and Singh, S. P. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.* 112(1-2):181–211.

Tenenbaum, J. B.; de Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290:2319–2323.

Thrun, S., and Pratt, L., eds. 1998. *Learning to learn*. Kluwer Academic Publishers.

Thrun, S. 2002. Robotic mapping: A survey. In Lakemeyer, G., and Nebel, B., eds., *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann. to appear.

Weng, J.; McClelland, J.; Pentland, A.; Sporns, O.; Stockman, I.; Sur, M.; and Thelen, E. 2001. Autonomous mental development by robots and animals. *Science* 291:599–600.

Weng, J. 2004. Developmental robots: theory and experiments. *International Journal of Humanoid Robotics* 15(9).

Xing, E. P.; Ng, A. Y.; Jordan, M. I.; and Russell, S. J. 2002. Distance metric learning with application to clustering with side-information. In *NIPS*, 505–512.

Yu, C., and Ballard, D. H. 2004a. On the integration of grounding language and learning objects. In *AAAI*, 488–493. AAAI/MIT Press.

Yu, C., and Ballard, D. H. 2004b. A multimodal learning interface for grounding spoken language in sensory perceptions. *ACM Trans. Appl. Percept.* 1(1):57–80.

Yu, C.; Ballard, D. H.; and Aslin, R. N. 2003. The role of embodied intention in early lexical acquisition. In *25th Annual Meeting of Cognitive Science Society (CogSci 2003)*.