Machine Learning for Robots: A Comparison of Different Paradigms

Sridhar Mahadevan
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
mahadeva@csee.usf.edu

Abstract

For robots to be truly flexible, they need to be able to learn to adapt to partially-known or dynamic environments, to teach themselves new tasks, and to compensate for sensor and effector defects. The problem of *robot learning* has been an intensively studied research topic over the last decade. In this paper we critically examine four major formulations of the robot learning problem: inductive concept learning, explanation-based learning, reinforcement learning, and evolutionary learning. We describe some well-known examples of systems that fit under each formulation, and discuss their strengths and limitations.

1 Introduction

The problem of robot learning is essentially one of getting robots to do tasks without the need for explicitly programming them. Programming robots is extremely challenging, for many reasons. Sensors on a robot, such as sonar, behave in a complex unpredictable manner in typical unstructured environments, such as a crowded office building. Thus, understanding sensors is not sufficient; one has to also model how they work in a particular task environment. To program a robot, the task has to decomposed into a sequence of very low-level operations, such as moving joint angles or wheels. There are no good high-level robot programming languages that we can rely on. Thus, for these reasons there is considerable interest in robots that can automatically learn to do tasks.

In this paper, we view robot learning as a special case of the general problem of *machine learning*. Machine learning is a subfield of artificial intelligence (AI), whose ultimate goal is to replace explicit programming by *teaching*. We define teaching broadly in this paper to mean any form of instruction, ranging from examples of the desired behavior, domain knowledge of the task, or even weak performance feedback. Teaching is usually less arduous and more effective than explicit programming. Consider the problem of instructing a human to operate an automated teller machine. We can show the person how to operate the machine by actually going through the motions ourselves. Clearly, this mode of teaching would be far more effective for robots also, rather than painstakingly programming the robot to operate the machine.

Machine learning research has studied many different types of learning [22]. Generally speaking, there are two types of learning: supervised and unsupervised. In the former, a teacher carefully selects examples for the learner, whereas in the latter the learner is given little or no feedback on the learning task. In this paper, we will discuss two supervised paradigms, inductive concept learning and explanation-based learning, and two unsupervised paradigms, reinforcement learning, and evolutionary learning. We focus on these four paradigms for several reasons. Each of these paradigms has been the focus of interest in machine learning for at least a decade, and is at a fairly mature state. Furthermore, each of these paradigms has been applied to the problem of robot learning. Finally, these paradigms make a nice contrast since they greatly differ in how they approach the problem of robot learning.

It might be useful before outlining the rest of this paper to mention some limitations of this paper. This paper is not intended as a survey of the field of robot learning. Much better sources exist for this purpose. There are a number of books and edited collections on robot learning, including Connell and Mahadevan [4], Dorigo [9], Franklin et al. [10], and Mahadevan [19]. This paper also is not intended as a survey of the different machine learning paradigms. For that, the reader is referred to a standard textbook on machine learning. The recent book by Mitchell [22] provides a comprehensive introduction to the field.

Here is a roadmap to the rest of this paper. Section 2 discusses the sorts of knowledge that robots might need to learn, and why the robot learning problem is difficult. Section 3 attempts to abstractly define the problem of robot learning as a three-dimensional credit assignment task. The different learning paradigms are subsequently described in Section 4. Finally, Section 5 compares these paradigms based on how they address the credit assignment problem, and discusses a challenge problem that might lead to a machine learning paradigm.

2 What Things should Robots Learn?

Before delving into the problem of robot learning, it might be instructive to briefly discuss the sorts of knowledge that would be useful for robots to learn. We can distinguish at least three types of such knowledge.

- *Control knowledge:* In any given task, there is usually some knowledge that we can easily provide the robot, such as action primitives. However, there is also knowledge that would involve a lot of human effort. For example, it is often difficult to put together a sequence of action primitives to achieve a given goal. In such situations, it would be desirable for the robot to learn which action to perform in any given situation, i.e. a control program.

- *Environment Models:* One benefit of designing robots is that they can be used in unknown environments, such as Mars or the ocean floor. Clearly, in such situations it is imperative that the robot be able to learn a map of the environment by exploring it. Also, most unstructured task environments are *dynamic*: objects move around from place to place, and actions can alter the enviroment in unpredictable ways. Even if the robot was given an accurate and complete model of the environment to begin with, this knowledge could quickly become useless. Thus, it would be beneficial if a robot could constantly update its knowledge of a given environment.

- *Sensor-Effector Models:* It is very useful for a robot to learn models of its sensors and effectors. For example, sonar models can be used to represent spatial information using evidence grids [26]. Although it is possible to use idealized sensor models, such as gaussian profiles, the advantage of learning sensor models is that this can be fine-tuned for a particular task environment [35]. Effector models are also very useful for the robot to learn, since for example they can be used to solve the inverse kinematics problem of controlling a high degree-of-freedom robot arm.

From the different sorts of knowledge that a robot has to learn, it seems clear that supervised learning alone cannot get us very far. For example, to learn in an unknown environment, the robot has to move around it in an exploratory trial-and-error fashion. Thus, robot learning must necessitate both supervised as well as unsupervised learning.

2.1 Why is Robot Learning Hard?

So far, we have discussed the types of information that robots should be capable of acquiring. This leads immediately into the next question: why is robot learning a difficult machine learning problem? We hinted at some reasons in the introduction. Now we can give a more detailed set of reasons for why robot learning is a challenging problem.

- *Sensor noise:* Most robot sensors, including inexpensive devices such as sonar as well as very expensive systems such as laser range scanners, are unreliable. Such transducers sometimes fail to see an object, or alternatively misjudge its distance. Thus, state descriptions computer from robot sensors are bound to be inaccurate to some degree.

- *Stochastic actions:* Due to sensing errors as well as due to the inherent complexity of the real world, robot actions may rarely appear to be deterministic. For example, if the robot picks up an object, the object may slip and fall sometimes, and other times the action may be successful.

- *Real-time response:* Robots must be capable of reactive planning, that is a robot must respond to unforeseen circumstances in real time. For example, a robot operating in an office environment must be ever alert to the possibility of collisions from some unanticipated obstacle in its path, ranging from people moving randomly around to junk left around in coridoors.

- *Online learning:* The training data for teaching a robot may not be available offline. Consequently, a robot may be required to explore its environment to collect sufficient samples of the necessary experience. Incrementality is a desirable trait for any robot learning algorithm, since the training data will be acquired only over time.

- *Limited training time:* A robot cannot have the luxury of training for months of real-time, although such extended runs are common in simulated systems (e.g. [34]). For a learning algorithm to be effective on a real robot, it must be able to produce satisfactory results from training experience that can be collected in a few hours or less (although the data collected could be processed offline for a much longer time, if necessary).

- *Situated Representations:* A robot becomes aware of its environment primarily through its sensors. Thus, a learning algorithm must be able to work with the limitations of sensors used. For example, a navigation robot may not be able to sense its exact coordinate location on a map, and must deal with this localization problem.

All of the above factors conspire to make the robot learning problem extremely difficult. However, this fact has not stood in the way of some impressive achievements in this field, as we will see when we discuss some of the major approaches studied in the literature. Before describing the specific paradigms, it is useful to attempt to provide a general definition of the robot learning problem.

3 A 3-D Credit Assignment Problem

In this section we present a general characterization of the robot learning problem. Essentially, we view the robot learning problem as one of learning a *policy* function π from some set of sensory states S to some set of actions A. This characterization is similar to other characterizations in the literature (for example, see Nehmzov and Mitchell [28]). The states and actions can be discrete or continuous. Examples of policy functions include control behaviors for mobile robots, such as avoiding obstacles, following walls, and moving a robot arm to pick up some object. The policies can be *stationary*, that is the mapping is a time-invariant function, or it can be *non-stationary*. The problem of learning policy functions generally involves solving a three-dimensional credit assignment problem, as shown in Figure 1. The *temporal* credit assignment problem is one of deciding which actions in a (possibly very long) sequence of actions is responsible for a good (or bad) outcome. The *structural* credit assignment problem involves determining the range of sensor values which would result in a similar outcome. Finally, the *task* credit assignment problem is to generalize the action sequence itself to accomplish other similar tasks.

As we will see below, each of the paradigms we present takes a different approach to solving this three-dimensional credit assignment problem. The first paradigm, *inductive concept learning*, assumes that a teacher presents examples of the target function for the robot. In this paradigm, the temporal credit assignment problem is non-existent, since the teacher is essentially telling the robot what action to perform, in some particular situation.

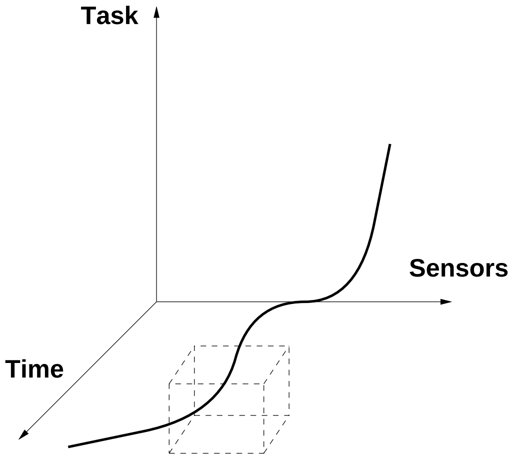

Figure 1: Robot learning involves solving a three-dimensional credit assignment problem.

However, the robot must still figure out the range of sensory states over which the same action is useful, so the structural credit assignment problem remains. Finally, the task credit assignment problem also has to be solved, since the examples provided by the teacher pertain to some particular task.

In the second paradigm, *explanation-based learning*, the teacher not only supplies the robot with example of the target function, but also provides a domain theory for determining the range of sensory situations over which the example action is useful. Thus, in this paradigm, the learner gets substantial help with both the temporal and the structural credit assignment problem. Note that the domain theory may be imprecise, which means that the robot has to be able to incrementally refine the theory using examples.

In the third paradigm, *reinforcement learning*, the teacher only provides the robot with a very weak scalar feedback signal. Not only is the robot not told which action to perform, but also the feedback could be "delayed" until some goal is achieved. Thus, in this paradigm, all three credit assignment problems must need to be solved.

The last paradigm, *evolutionary learning*, is very similar to reinforcement learning, in that the robot is only provided with a scalar feedback signal. However, there are several major differences between this paradigm and the previous one, both in terms of how learning is achieved (online vs. offline), as well as how the space of solutions is searched (greedy vs. non-greedy).

4 Four Machine Learning Paradigms

4.1 Inductive Concept Learning

We begin with the most classical paradigm in machine learning, namely inductive concept learning. Figure 2 characterizes the problem formulation covered by this paradigm. The robot knows that the policy π it is learning comes from some space of hypotheses H. The

- **Given:**
    1. A space of hypotheses H, each of which describes a policy function $\pi : S \rightarrow A$ over some set of states S and actions A.
    2. A set of training examples E of the target policy π^*, sampled using a (fixed but unknown) probability distribution P on the underlying instance space I.
- **Determine:** A policy $\pi \in H$ that minimizes the expected error with respect to the target policy, measured over I using distribution P.

Figure 2: Inductive Concept Learning paradigm.

robot is also provided with a set of training examples E of the target policy. These examples are drawn from some space of instances I according to some unknown but fixed probability distribution P. Note that this formulation of the inductive concept learning problem is based on the *Probably Approximately Correct* (PAC) model [27, 37]. The goal of the robot is to determine a policy π in H that minimizes the expected error

$$\sum_{i \in I} Prob(\pi(i) \neq \pi^*(i))$$

Many algorithms have been developed for solving the inductive concept learning problem, but two of the most well-known algorithms are decision trees [31] and neural nets [21]. A classic example of a robot learning system based on the inductive concept learning paradigm is Pomerleau's ALVINN system [29]. Here, a neural net is trained using backpropagation to drive a real Ford truck on several different types of road terrain, ranging from regular highways to dirt tracks. The training data for ALVINN consists of pairs of camera images of the road and steering commands. The data is collected by having a human drive the truck. One of the most impressive features of ALVINN was that it learned to drive very quickly, in most cases in less than 10 minutes of real-time. A key reason for this fast convergence is that for each human-supplied training example, ALVINN synthesized dozens of new examples by scaling and rotating the input image. For each such transformation, ALVINN used a steering model to determine the corresponding new action output. Also, ALVINN used some clever tricks to prevent the net from overfitting the training data, such as making sure that the training examples in the buffer were sufficiently diverse. There are many other examples of inductive concept learning applied to robot learning [4, 9, 10, 19].

Since the robot is told what action to perform, the inductive learning paradigm does not require solving the temporal credit assignment problem. However, the robot does need to generalize the range of sensory states where the recommended action can be performed. Neural nets and decision trees are two ways of solving this structural credit assignment problem. Inductive concept learning does not specifically address the task credit assignment problem of transferring learning across related tasks. Thrun and Mitchell [36] propose an interesting

- **Given:**

  1. A space of hypotheses H describing policy functions $\pi : S \rightarrow A$ over some set of states S and actions A.
  2. A domain theory D describing the target function π
  3. An *operationality criterion*, which constrains the representation of the learned policy.
  4. A set of training examples E of the target policy π^*, sampled using probability distribution P on the underlying instance space I.

- **Determine:** A policy $\pi \in H$ whose description satisfies the operationality criterion, and is consistent with D and minimizes the expected error with respect to the target policy π^*.

Figure 3: Explanation-based Learning paradigm.

lifelong learning approach based to extend the inductive concept learning framework to handle task transfer. Their approach is based on finding *invariances* across related functions. For example, given the task of recognizing many objects using the same camera, invariances based on scaling, rotation, and image intensity can be exploited to speed up learning. The ALVINN system described above can also be viewed as exploiting invariances (in this case, of a driving function) to accelerate learning.

The principal weakness of the inductive learning paradigm is that it requires an experienced teacher, who can supply a diverse enough set of training examples. Collecting this training data can be difficult in some domains; for example, a robot sent to explore Mars or the ocean floor could not assume that it would be given an adequate training set of examples. Another problem with the inductive learning paradigm is that function approximators, such as neural nets, often taken thousands if not more examples to converge. One critical issue is how to speed up learning by incorporating some form of *bias*. Both the examples cited above used some kind of invariance property to bias learning. In the next paradigm, we discuss how general forms of bias can be used to accelerate learning.

4.2 Explanation-based Learning

Humans appear to be capable of making generalizations from a few examples. Clearly, humans bring to bear considerable amount of background information to the learning task. The paradigm of *explanation-based learning* (EBL) [7, 23] studies how domain knowledge about the function being learned can be used to speed up learning. This formulation is described in Figure 3.

EBL does not assume any particular representation for the domain theory. It can be a logical theory, or a neural network, or even an approximate qualitative physics based theory.

Some examples will illustrate the diversity of possible approaches. Dejong [6] describes an algorithm called EBC (explanation-based control) which learns to control a simulated robot arm. This task is highly non-linear and poses a difficult control problem for classical approaches. The approach used in EBC is to apply a qualitative physics based domain theory to "explain" positive examples of the desired behavior. In this case, the problem was to swing the robot arm upwards, which is difficult since the arm is weak and cannot be directly lifted up. EBC constructs a recursive explanation of the given behavior, and optimizes it to produce a workable algorithm for swinging the arm.

Another example of the EBL paradigm is the EBNN algorithm proposed by Mitchell and Thrun [24]. In their system, the domain knowledge is a (approximate) model of each action represented by a neural net. These nets are trained using examples gathered during a previous learning task. The action model nets can be used to analyze sample state trajectories where the robot successfully achieves its goal. Since the action model is represented as a differentiable feedforward neural net, it is possible to extract the slope of the function represented by the net, and use this information to speed up learning the policy.

The primary advantage of the EBL formulation of the robot learning problem is that it provides a way of incorporating previous domain knowledge or bias to speed up learning. The savings in number of examples needed to learn the policy can be quite significant. However, the main requirement is that for this approach to succeed, the robot needs to be given (or have previously learned) some domain knowledge. Also, if the domain theory is very approximate, the robot needs to exercise some care in using the theory to guide its generalization. Much work remains to be done in addressing these issues.

4.3 Reinforcement Learning

Reinforcement learning (RL) studies the problem of inducing by trial and error a policy that maximizes a fixed performance measure (or reward) [2, 13, 33]. RL is an unsupervised paradigm, meaning that examples are not carefully selected by a teacher. Instead, the distribution of examples is influenced by the robot's actions, since the states and rewards experienced by the robot depends on the actions it takes. In this paradigm, the robot is faced with a difficult temporal credit assignment problem of evaluating the goodness of states and actions from a scalar reinforcement signal, not a very helpful explanation! Figure 4 describes the RL paradigm.

RL has several nice properties. It does not require supplying the robot with a theory of its domain, as is required by EBL, which might be a substantial undertaking in any real world robotics task. Second, RL can be used for *online* learning, as opposed to some types of inductive learning, thus the robot is continually improving its performance as it learns. Third, supplying the robot with suitable reward functions turns out to be relatively straightforward, for many tasks. On the other hand, reinforcement learning suffers from a number of limitations. It can be very slow, requiring millions of steps to converge. It is also difficult to incorporate domain knowledge to speed up learning.

One example of work that addresses both these issues is that of Mahadevan and Connell [20]. They conducted a detailed experimental study of using RL to train a real robot to do a

- **Given:**
  1. A space of hypotheses H describing policy functions $\pi : S \rightarrow A$, where S is the set of states, and A is the set of actions.
  2. A reward function $r : S \times A \rightarrow \mathcal{R}$.
  3. An optimality criterion O that maps any policy π and reward function r to a value function $V_r^\pi : S \rightarrow \mathcal{R}$.
- **Determine:** An *optimal* policy π^* that results in a maximal value function V^* over all other policies.

Figure 4: Reinforcement Learning paradigm.

box-pushing task to a level superior to a handcoded program. The key idea underlying their work was to decompose the overall task into several simpler behaviors – find a box, push a box, or unwedge from a stalled state. This decomposition was provided to the robot, and not learned. Each of these behaviors was learned using Q-learning [38], a well-known model-free RL algorithm. Q-learning solves the temporal credit assignment task by propagating delayed rewards backwards across actions, in an incremental fashion. Since the robot was faced with learning a policy over hundreds of thousands of sensory states, it faced a difficult structural credit assignment problem. A statistical clustering algorithm was used to address this problem. Other researchers, including Lin [16] and Dorigo [8], have also demonstrated the benefits of using a behavior-based decomposition of the learning task, although their systems differ in other respects. Lin's system used a neural net to address the function approximation problem, and Dorigo's system uses genetic algorithms. Both these systems have also addressed the problem of coordinating multiple behaviors. Asada [1] describes a vision-based robot that uses a behavior-based decomposition to learn a soccer task.

The RL paradigm is quite appealing for training robots. It is an online learning paradigm, and does not require supplying the robot with either training examples or any background knowledge beyond a scalar reward function. However, the RL paradigm has some key limitations. Learning can be very slow, unless considerable background information is pre-specified. Also, RL assumes that the environment can be modeled as a *Markov decision process* [30], which implies that the current state and the action determines a probability distribution on future states. This "memoryless" property is clearly false, since robots can rarely sense the true state of their environment. There is growing interest in extending RL to partially-observable MDP's [17, 32]. In particular, Koenig and Simmons [14] describe a map-learning algorithm that is based on partially-observable MDP's. At present, these algorithms do not appear to be tractable for very large state spaces. Connell and Mahadevan [4] describe a heuristic algorithm called Q-maps, which extends Q-learning to probabilistic evidence grids [25, 26], which is a very high-dimensional partially observable state space. The key idea here is to express the utility of a state as a weighted sum of the utilities of each cell in the evidence grid. Cassandra et. al [17] discuss a variety of algorithms for solving partially-observable

- **Given:**
  1. A space of hypotheses H describing policy functions $\pi : S \rightarrow A$, where S is the set of states, and A is the set of actions.
  2. A fitness function $f : \pi \rightarrow \mathcal{R}$.
  3. An encoding E mapping policies π to a binary bit string representations.
  4. A set of genetic operators $O : H \rightarrow H$ that transform policies π.
- **Determine:** An *optimal* policy π^* that maximizes the fitness function f.

Figure 5: Evolutionary Learning paradigm.

MDP's, including several refinements of the Q-maps algorithm.

4.4 Evolutionary Learning

The second unsupervised paradigm we will discuss is evolutionary learning, which includes genetic algorithms [11] and genetic programming [15]. This paradigm is described in Figure 5. Although this paradigm is similar in some respects to the reinforcement learning paradigm, there are significant differences. The first difference is in the search strategy: evolutionary learning starts with a *population* of policies, and combines them to produce better policies till an optimal policy is found. The combination is achieved through genetic operators, such as crossover and mutation. Ech policy is encoded using some standard bit string representation. Note, however, that in genetic programming, a higher level representation using LISP s-expressions is used. The fitness function is some measure of the goodness of a given policy.

A number of researchers have studied using evolutionary learning to train robots, including Dorigo [8] and Grefenstette and Schultz [12]. The common framework in these systems is the use of a *classifier system*, which is a production-rule type system, a strategy for conflict resolution among competing rules, and an algorithm for assigning credit to individual rules. The credit assignment rule used is the *bucket brigade* algorithm. These systems have been used to train robots to avoid obstacles, to navigate around rooms, and learn goal-seeking behaviors like find bright lights. Davidor [5] describes the use of genetic algorithms to design robot arms.

The evolutionary learning paradigm has some important strengths, including the ability to start with a good set of policies. This allows the designer to start the system in a "primed" state, which helps speed learning. Another advantage of the evolutionary learning paradigm is that it is not constrained to stationary policies, but can learn arbitrary policies. However, the convergence issues are unclear for general policies.

A key weakness of the evolutionary paradigm is that it does not easily allow for online learning. All training must be done on a simulator, and then the learned policies can be tested on a real robot. In practice, this limits this approach to those tasks where a good simulation is available. However, designing a good simulator for the general problem of mobile robots operating in unstructured environments, such as a crowded office or lab, using high-dimensional sensors such as vision, is an enormously difficult task.

5 Discussion

We now summarize the four learning paradigms in Table 1, according to how they address the credit assignment problem. In the inductive learning paradigm, the temporal credit assignment problem is solved by the teacher. The structural credit assignment is solved using some function approximation algorithm, such as a decision tree or a neural net. The inductive learning paradigm does not specifically address the task credit assignment problem, although extensions such as the lifelong learning approach of learning domain invariances appear very promising. In explanation-based learning, both the temporal and the structural credit assignment problem are solved by the teacher, who provides both examples and a domain theory for safely generalizing the examples. One way to address the task credit assignment problem is through learning action models, such as done in the EBNN algorithm. In the reinforcement learning paradigm, the temporal credit assignment problem is addressed through using an algorithm like Q-learning. The structural credit assignment problem is addressed through using some standard function approximator. The task credit assignment problem can be addressed by learning general action models [18]. Finally, the evolutionary learning paradigm addresses the temporal credit assignment problem through using the bucket brigade algorithm. The structural credit assignment problem is addressed through the genetic operators for transforming policies, whereas the classifier system framework deals with the problem of transferring learning across multiple tasks.

Paradigm	Temporal	Structural	Task
Induction	Teacher	Function approximator	Invariances
EBL	Teacher	Domain Theory	Domain Theory
RL	Q-learning	Function approximator	Action models
Evolution	Bucket brigade	Genetic operators	Classifier Systems

Table 1: Comparing different robot learning paradigms based on how they address the credit assignment problem.

Robot learning is one of the most interesting and difficult machine learning problems. While much progress has been made by many researchers using different paradigms, much more remains to be done in scaling up the algorithms to work with high-dimensional sensors such as vision, to handle partially observable states, to deal with continuous actions, and to deal with learning from limited number of examples. A good way to conclude this look at robot learning is to propose a challenge problem for robot learning. An excellent choice for such a challenge problem comes from the COG humanoid robot being developed at MIT [3]. This system raises a number of very fundamental learning issues in integrating different modalities, such as vision, action, language, and cognition. It is quite possible that in attempting such an integrated system, we will uncover a completely new machine learning paradigm.

Acknowledgements

The author's work in robot learning is supported, in part, by a National Science Foundation (NSF) CAREER Award Grant No. IRI-9501852. Parts of this paper were derived from discussions with Jonathan Connell, during the preparation of an edited book on robot learning [4]. I thank Sebastian Thrun for his detailed comments on a draft of this paper.

References

[1] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Purposive behavior acquisition for a real robot: Vision-based reinforcement learning. *Machine Learning*, 23(2-3), 1996.

[2] A. Barto, R. Sutton, and C. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):834–846, 1983.

[3] R. Brooks and L. Stein. Building brains for bodies. *Autonomous Robots*, 1(1):7–25, 1994.

[4] J. Connell and S. Mahadevan. *Robot Learning*. Kluwer Academic Publishers, 1993.

[5] Y. Davidor. *Genetic Algorithms and Robotics*. World Scientific, 1991.

[6] G. Dejong. A case study of explanation-based control. In *Proceedings of the 12th International Conference on Machine Learning*, pages 167–175. Morgan Kaufmann, 1995.

[7] G. Dejong and R. Mooney. Explanation-Based Learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.

[8] M. Dorigo. ALECSYS and the AutonoMouse: Learning to control a real robot by distributed classifier system. *Machine Learning*, 19(3), 1995.

[9] M. Dorigo. Introduction to the special issue on learning autonomous robots. In M. Dorigo, editor, *IEEE Transactions on SMC*, volume 26, pages 361–364. June 1996.

[10] J. Franklin, T. Mitchell, and Thrun. S., editors. *Robot Learning*. Number 2-3. Kluwer Academic Press, 1996. Special Issue of Machine Learning Journal, Vol 23.

[11] D. Goldberg. *Genetic Algorithms.* Addison-Wesley, 1989.

[12] J. Grefenstette and A. Schultz. An evolutionary approach to learning in robotics. In *Proceedings of the 1994 Workshop on Robot Learning*, 1995. Held in conjuction with the 11th International Conference on Machine Learning (IMLC '94). Online proceedings available at http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/robot-learning.html.

[13] L. Kaelbling. *Learning in Embedded Systems.* MIT Press., 1993.

[14] S. Koenig and R. Simmons. Passive distance learning for robot navigation. In *Proceedings of the 13th International Conference on Machine Learning*, pages 266–274. Morgan Kaufmann, 1996.

[15] J. Koza. *Genetic Programming.* MIT Press, 1993.

[16] L. Lin. *Reinforcement Learning for Robots using Neural Networks.* PhD thesis, Carnegie-Mellon Univ, 1993.

[17] M. Littman, T. Cassandra, and L. Kaelbling. Learning policies for partially-observable environments: scaling up. In *Proceedings of the 12th International Conference on Machine Learning*, pages 362–370. Morgan Kaufmann, 1995.

[18] S. Mahadevan. Enhancing transfer in reinforcement learning by building stochastic models of robot actions. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 290–299. Morgan Kaufmann, 1992.

[19] S. Mahadevan. Proceedings of the workshop on robot learning. Held in conjuction with the 11th International Conference on Machine Learning (IMLC '94). Online proceedings available at http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/robot-learning.html, July 1994.

[20] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1992.

[21] J. McClelland and D.(editors) . Rumelhart. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Bradford Books, 1986.

[22] T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[23] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-Based Generalization: A unifying view. *Machine Learning*, 1(1), 1986.

[24] T. Mitchell and S Thrun. Explanation-based learning: A comparison of symbolic and neural net approaches. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 197–204. Morgan Kaufmann, 1993.

[25] H. Moravec. Certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

[26] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 116–121, 1985.

[27] B. Natarajan. *Machine Learning: A Theoretical Approach*. Morgan Kaufmann, 1991.

[28] U. Nehmzov and T. Mitchell. A prospective student's introduction to the robot learning problem. Technical Report UMCS-95-12-6, University of Manchester, 1990.

[29] D. Pomerleau. Neural network based autonomous navigation. In C. Thorpe, editor, *Vision and Navigation: The CMU Navlab*. Kluwer Academic Publishers, 1990.

[30] M. Puterman. *Markov Decision Processes: Discrete Dynamic Stochastic Programming*. John Wiley, 1994.

[31] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[32] S. Russell and R. Parr. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the Fourteenth IJCAI*, 1995.

[33] R. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224. Morgan Kaufmann, 1990.

[34] G. Tesauro. Practical issues in temporal difference learning. In R. Sutton, editor, *Reinforcement Learning*. Kluwer Academic Publishers, 1992.

[35] S. Thrun, A. Bucken, W. Burgard, D. Fox, T. Frohlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, P. Bonasso, and Murphy. R., editors, *AI-based Mobile Robots: Case-studies of Successful Robot Systems*. MIT Press. To appear.

[36] S. Thrun and T. Mitchell. Learning one more thing. Technical Report CMU-CS-94-184, Carnegie-Mellon University., 1994.

[37] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11), 1984.

[38] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England, 1989.