Implementing a Learning System for Subsumption Architectures

John Shewchuk
Paul Viola

Technical Report No. CS-89-17

February 1989

Implementing a Learning System for Subsumption Architectures

John Shewchuk*and Paul Viola†
IBM, T.J. Watson Laboratory
Yorktown Heights, New York

March 9, 1989

Abstract

We have built a robot that wanders about the environment and learns from novel experiences. The robot learns to perform those actions that bring about a pre-determined desirable state. We argue that some of the difficulties associated with learning are reduced by embedding those modules capable of learning within a subsumption architecture.

The claim that subsumption architectures are well suited to the creation of real-time robust control systems is not new. However, until now little has been said regarding the incorporation of learning. We claim, and hope to demonstrate, that an augmented subsumption architecture is especially well suited to the task of learning behaviors.

We have found that a simple learning algorithm guided by a utility function can enable a robot to discover and exhibit new behaviors. In addition, by calculating a certainty factor associated with every learned response the robot can moderate the effects of learning on its behavior.

1 Introduction

We have built a robot that functions and learns in real environments. The world is a complex and dynamic place; coping with this dynamic complexity requires a robust control scheme that can adapt to aspects of the environment that can change or are unknown. We have chosen to control our robot

*Currently at Brown University.
†Currently at MIT.

with a subsumption architecture that incorporates a learning mechanism. The subsumption architecture makes it easy to create a simple and robust robot quickly. Additionally, it is a computational model that explicitly facilitates the subsumption of precompiled behavior by learned behavior.

Rather than address the general problem of arbitrary robot behavior we have instead chosen the smaller class of reactive behaviors. Our robot's control program contains no world state. In support of this controversial decision we cite the arguments of Agre [2] and others, who claim that reasonably complex and directed behaviors can be implemented by computing a function of current sensor readings. We feel justified in this both because reactive systems are an open area of investigation and because it has reduced the initial complexity of our explorations.

To learn new behavior the robot observes its own dynamic interaction with the environment. In doing so, it attempts to decide when a particular reaction to a *situation* has allowed it to achieve a desired state (or fall into an undesired one). Clearly, if an identical situation arises in the future, this observed reaction would yield the same successful/unsuccessful results. By generalizing, one concludes that a robot should repeat "successful" reactions in similar situations that arise in the future.

An action is evaluated by examining the context in which it is taken and by examining results of taking it. We call this evaluation procedure a *utility function* and the value the function takes *utility*. Metaphorically, the utility function models the robot's own sense of good and bad or pain and pleasure. Actions in context that yield high utility are good or bring pleasure. Actions in context that yield low utility are bad or bring pain.

This has obviously strong implications for any learning system.

When an action has high utility, the robot stores an association between the sensor reading that describes the situation and the action. With time, the robot collects a quantity of these associations. They make up the robot's constantly developing beliefs about the behavior of the world. These beliefs are fundamentally dependent both on the robot's concepts of pain and pleasure and the dynamics of its interaction with its environment.

As the robot grows confident in its beliefs regarding particular situations, it begins to incorporate them into its behavior. This incorporation is self-reinforcing. When acting on beliefs yields "good" results, beliefs are reinforced. In time, those modules of the robot responsible for formulating beliefs begin to assert a stronger and stronger influence over the robot's choice of action.

We will discuss in detail our implementation of these concepts on a mo-

bile robot. In an attempt to remain concrete, we first describe the robot: its geometry and physical attributes. We then the describe the structure of the controlling subsumption architecture; in doing so we address its suitability for incorporating learning. We then address the learning mechanism, focusing on its interaction with the world and the subsumption architecture. Simple examples and experimental results follow. Finally, we analyze the learning mechanism, its behavior and theoretical limitations.

2 The Robot

This section describes the components and techniques used to assemble the robot. The robot's mobile base is a Transitions Research Labmate upon which we have mounted 8 radial sonars and 4 forward facing infra-red detectors. The sensors are controlled by a Transitions Research proximity subsystem. The base and sensors are linked to a Symbolics 3640 that runs a subsumption architecture. The Symbolics is linked, by a local network, to a Sun 4 which runs the learning system.

The sensory input to the subsumption architecture is a data structure containing an 8 element sonar vector, a 4 element infra-red vector, the robots current velocity, and its rate of turn[1]. The effector commands sent to the robot base control velocity and rate of turn. The effector commands specify the derivatives of these variables[2].

2.1 Subsumption Architecture

The subsumption architecture control program for the robot is comprised of three levels[3]. Figure 2.1 depicts our subsumption architecture graphically[4].

- The first level wanders. To a first approximation it makes the robot move about the room randomly. To do this, the rate of forward motion and the rate of turn are varied. We have experimented with alternative mechanisms for wandering. They vary in their level of robustness

[1]The resolution of each of these variables is on the order of 8 bits.

[2]There are several discrete velocity and turn rate derivative messages available. The velocity messages are: stop (decelerate a large percentage of current velocity per time unit), accelerate, decelerate, and non-committal (implying that the sender is unconcerned with velocity). The turn rate messages are similar.

[3]For a complete and thorough description of a subsumption architecture see [4, 3]

[4]This format for depicting Subsumption Architectures is due to Brooks et. al.

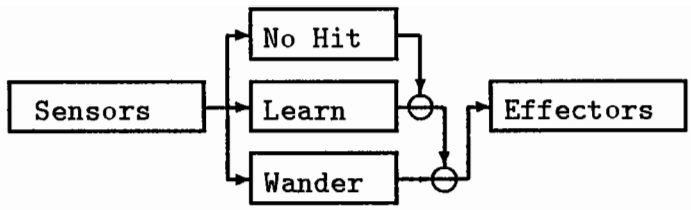

Figure 1: Subsumption architecture control program for the robot.

(mean time between failure), and exploratory power. Since the learning mechanism relies on this lowest level to present it with a variety of experiences, we chose a low level that seemed to get into the most varied set of situations.

- The second level learns. It observes the behavior of the robot and attempts to learn reactions that will yield high utility.
- The third level avoids collisions. This level plays an especially critical, though rare, role in the behavior of the robot. By avoiding the very worst situations it guarantees that the learning mechanism, in a fit of confusion, will not be able to command lethal action. It also serves to censor experiences. If all goes well, the learning mechanism will never have the opportunity to mislearn from bad examples.

The levels are independant paths from senses to effectors. These paths compete for finite effector resources. The competition is resolved by a subsumption network that is most easily described as a total ordering. The ordering is (from lowest priority to highest) wander, learn, and no-hit. Levels are designed so that they compete for effectors only when they can reliably assess a situation. This leads to overall robot behavior that is as robust as the most robust level and (whenever possible) as complex as the most complex level.

Our version of the subsumption architecture differs from those described in the past. Previous implementations use a form of positional control in which sensor data is processed frequently yet the robot is commanded to move large distances (that require a large amount effect). In these robots lower levels diligently examine the world in order to prevent collisions that may result in a dynamic world.

We claim that while sensitive to changes in the world, and therefore robust, earlier subsumption architectures are not "dynamic" enough. Our implementation reads sensor values, assesses the current situation, calculates effector commands and learns 10 times per second[5]. This has enabled us to adopt a form of derivative control that provides for smoother and more aesthetic motion. The robot calculates the change in velocity and rate of turn at any given time. The more frequently a system runs, the less a single erroneous sensor reading will effect the behavior of the robot and the more successfully arbitrary spatial constraints can be handled. As a result the robot is robust in the face of non-repeatable sensor errors or spurious messages from higher levels.

2.2 Continuous Derivative Control

A reactive robot can never determine if a single sensor reading is in error or misleading; there is no context to which an error can be compared. Consequently a sequence of effector calculations may differ drastically in their conclusions. Similarly, a learning system may believe that the course of action that is called for in a particular situation should differ drastically from that currently being followed. However for a real robot, one governed by the real physical restrictions of momentum and finite available energy, drastic changes of action are not only expensive but often impossible. This problem is only compounded by non-repeatable sensor error and the naivete of a learning system in its early stages of experience.

Continuous derivative control is a physically realistic scheme that provides for a smooth integration of belief changes into the behavior of the robot. For example, an accurate reappraisal of a situation may dictate a change from a stationary state to a state where the robot is moving forward and not turning. If the learning system is confident that such a change is required, yet repeated infrequent sonar errors cause the system to intermittently believe otherwise, our robot control architecture will simply move forward. If however an overwhelming number of sensor readings confidently imply that the robot should turn to the right and be stopped, the robot will controllably move from the previous course of action to this one. This robust behavior in the face of rapid and possibly inaccurate changes in belief is a very desirable platform on which to build a learning system.

[5]Much work has gone into the design and implementation of a particularly efficient subsumption architecture, see [Viola] for a partial description.

2.2.1 Learning and the Subsumption Architecture

We claim that the subsumption architecture is especially amenable to the addition of a learning algorithm.

- It allows a robot to safely contend with the outputs from a learning system that can make mistakes.
- It provides a robot with a layer of robust behavior that is retained throughout the learning process. The learning mechanism can rely on this layer whenever it is unsure of the appropriate action in a particular situation.

The subsumption architecture provides our learning system with a variety of experiences while ensuring that the robot operates safely. More conventional learning techniques place the naive learning mechanism into an environment where it has primary responsibility for control. It is difficult to imagine such a scheme operating on an actual robot where the result of an incorrect action can be disastrous.

2.3 The Learning System

Our learning paradigm creates associations between sensor encodings and effector encodings. The strength of the association is determined by the number of learning examples the robot encounters and by the utility of the sensor data when the association was learned. From every learned reaction a certainty factor can be calculated. These certainty factors provide a mechanism for ignoring erroneous or unlikely associations. In section 4.2 we describe the learning algorithm in detail.

2.3.1 The Utility Function

The utility function moderates the strength of the associations between sensor and effector encodings. In this way it determines what response the learning system will suggest in a new situation. Designed properly, a utility function allows learning from a small set of examples, while not overwhelming the system with unnecessary associations.

Our current utility function is reactive; its value is only dependent on current sensor readings. If at least two of the three rear sonars register objects nearby and the rest of the sonars do not detect objects, utility is positive. Surprisingly, this simple function is sufficient to cause the robot to develop an affinity for walls and corners.

2.3.2 Certainty Factors

A level within a subsumption architecture is designed so that it will command an action unless it is reasonably sure that the results of that action are warranted. For example a collision avoiding level should not command actions in situations where there are no obstacles in sight; in these situations any avoidance would be unwarranted. Without this design restriction the actions taken by the robot would be completely dictated by the highest priority level. All subsumption architecture programmers use this as an implicit guideline for creating subsumption levels. Our learning mechanism must do the same.

Theoretically a learning system should be able to calculate a *certainty factor* associated with a computed reaction to a given situation. This certainty factor should be closely related to the probability that the system has correctly assessed the situation and that a resulting situation with high utility will ensue. Armed with this information the learning system should determine whether or not it actually wishes to send that effector command (running the risk of subsuming some more appropriate command).

2.3.3 Learning from Sensors and Effectors

As the robot moves about its environment sensor values and reactions are sent to the learning system. These values are encoded in a sensor/effector vector. Transforming raw sensor data to and from a form that is amenable to analysis by the learning algorithm is a difficult problem. For this implementation we decided to use a widely accepted encoding called a thermometer code. Unfortunately thermometer codes are not "distributed" codings and so are less likely to be orthogonal[6]. As an example of thermometer encoding figure 2.3.3 shows a vector containing three sonar readings and two effector outputs. The height of the bar is proportional to the activity of the unit in that position. This figure would indicate that there is an object nearby on the left, nothing ahead, and an object relatively distant to the right. The outputs indicate that the robot is turning slightly to the right and moving at high speed.

[6]We will see from the analysis in section 4.2 that this may prove to be an important limitation of our learning algorithm.

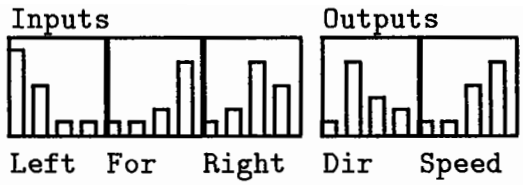

Figure 2: A Sample encoding. This composite vector describes 5 independant variables: the left sonar reading, the forward sonar reading, the right sonar reading, the direction of turn and the speed of forward motion. Each of the 5 component variables is encoded as a logarithmic thermometer code. The most appropriate interpretation of each of the components in turn is: close range, far range, medium range, turning to the left, and moving forward quickly.

3 Experimental Results

To test the techniques described in this paper, we designed a simple project for our system: the robot should keep its rear three sonar sensors near objects (causing them to register). By observing patterns of sensor-effector pairings while the robot wanders, the learning system builds an association between sensor readings and effector commands that cause the robot to keep its rear toward walls. When similar environmental situations arise, the learning system directs the robot to repeat those previously successful actions.

It is important to realize that the task of keeping the rear of the robot near a wall was neither implicitly nor explicitly implemented in the subsumption architecture. Prior to the learning episode, the robot would never come to a sustained halt with its rear toward an obstacle. If the robot had halted, the wander level would have caused it to wander away. This project was antithetic to the behavior of the robot before it had learned.

To begin the learning process we simply allow the robot wander without supervision. The robustness of the controlling subsumption architecture insures that the robot does not damage itself. The robot wanders about the room eventually encountering situations with high utility. It then stores an association between sensor readings and effectors commands (between situations and actions). Eventually the reaction associated with a situation

becomes sufficiently strong for the learning system to send a command to the subsumption architecture. As time passes the robot increasingly finds situations with high utility.

3.1 Performance

When the robot is first turned on, it wanders about the room in relatively straight though irratic paths. These paths routinely bring it toward walls and obstacles. These objects cause the robot to stop moving forward. The random component of the wander mechanism's turn commands eventually turn the robot away from the object, and it then wanders away. When the robot "sees" this obstacle to the rear the utility function begins to return a non-zero value. As a result the robot begins to learn from the actions it is taking.

This learning associates the current action with the current sensor reading. These actions are characterized by a low forward velocity while there is no characterization for turn rate (the turn rate is equally distributed between left and right). Figure 3.1 is a pictorial example of three of these situations. In the figure the rate of forward motion is low in each of the paths: the turn rate is low and to the left in the first path, low and to the right in the second, and high and to the left in the third. In linear combination these outputs result in an output strongly biased toward low speed but without firm commitment to a direction. This is exactly the reaction that achieves the goal we have set for the robot.

It takes about 10 minutes for our robot to develop a stop response to situations where its back is to a wall. Interestingly, while this response is quite strong, changes to the environment - such as the appearance of a person - will cause the learning system to cease generating the stop response. These situations are sufficiently different from any situation with high utility that the learning system is no longer confident regarding its beliefs. In these cases it refrains from intervening in the subsumption architecture's wander behavior. The subsumption architecture will then cause the robot to wander away.

In situations where the robot is not against a wall the learning system occasionally generates erroneous commands to be inserted in the subsumption architecture. These commands are spurious and do not occur in a statistically significant pattern. As we discuss above, such commands are handled gracefully by the derivative control mechanisms in the subsumption architecture.

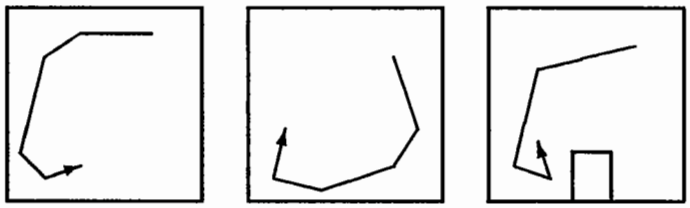

Figure 3: Sample paths

4 Analysis of the Learning Algorithm

This section describes how the the learning algorithm works and provides some analysis of its operation. A more detailed account of the algorithm and enhancements that can be made appears in[6].

The learning algorithm can be described as a linear associative memory. As such, its associations are susceptable to two distinct interactional phenomena:

- Actions associated with the same situations are combined by weighting each action by its utility and summing the results.
- Actions associated with seperate situations can effect each other if the situations are not linearly independant.

The first phenomena ensures that a frequently occuring response with high utility will become a large component of the response associated with a situation. In situations in which there is no clear best response the associated response will, by linear summation, look like noise. These phenomena are important for the proper associations to be formed.

The second is most closely related to generalization. We do not address its importance or its effects in this paper.

4.1 Algorithm

A simplified description of the algorithm proceeds as follows. We initialize the connection matrix, C, and the sensor and effector vectors, S and E. Next we enter the main loop. In this loop we read and normalize new sensor data and update S. Then we update the connection matrix using the

```
C ← 0
S° ← 0
E° ← 0
loop
    S ← Normalize(new sensors)
    C ← C + U(S)(ES°^T)
    E° ← CS
    CalculateAndSendCommand(E°)
    E ← Normalize(actual effects)
    S° ← S
endloop
```

Figure 4: Simplied version of learning algorithm

previous value of S and E modified by the utility of the current S. Using the updated connection matrix and S we generate effector commands that are sent to the subsumption architecture. Finally we read and normalize the actual response made by the robot and store this in E. This algorithm is shown in figure 4.

4.2 Operation of the Algorithm

In order to analyze this system, we first assume that all the elements of S, the set of all possible sensor encodings, are nearly orthogonal.[7] Therefore:

$$S_i^T S_j = \begin{cases} 1 & i = j \\ \epsilon & \text{otherwise} \end{cases}$$

Where S_i denotes a sensor vector at time i. Likewise E_i is an effector

[7]This requirement is not as restrictive as it may first appear since the probability that two vectors randomly chosen from an N dimensional space are nearly orthogonal approaches one as N grows. Also one can use sensor encodings that help ensure orthogonality; we are currently working on developing an approach we call semantic k-means to do this automatically. In the current robot we have had good results with the simple sonar encoding scheme described in section 2.3.3.

vector at time i. Let C_i denote C at time i. From the algorithm:

$$C_n = \sum_{i}^{n-1} U(S_{i+1}) E_i S_i^T$$

Consequently, $E_n = C_n S_n$ which is:

$$E_n = \sum_{i}^{n-1} U(S_{i+1}) E_i (S_i^T S_n)$$

Each term of this sum is a weighted inner product of the current sensor vector and the sensor vector terms stored in the associative memory. When the current sensor vector and one of the stored sensor vectors describe the same environment (their phase is the same) their inner product will be one, in all other cases it will be ϵ. Thus our reaction consists of two components:

$$\sum_{i \text{ in phase}} U(S_{i+1}) E_i + \text{ small cross-terms}$$

E_n is the linear sum of reactions to previous identical situations, sensor vectors, weighted by a utility function plus a small amount of noise. We need to interpret E_n as a command to the subsumption architecture. Unfortunately, because the magnitude of the reaction vector grows over time this interpretation is difficult. One interpretation not only deciphers the information in E_n it also associates a certainty factor with its value.

4.2.1 Interpreting E

It is important to realize that if the value of E_n is valid, its magnitude may grow but its phase remains essentially constant. It is the phase describes the relative activations of the effector encoding. If we use a simple gain control system and an encoding such as the one described in section 2.3.3 then we can ignore the magnitude of the output when interpreting E_n.

Since long outputs can sometimes represent consistent training or training with large utility, magnitude contains information about how certain we are of this particular output. As described in section 2.3.2, we can use this certainty to decide whether to send the command described by the phase. Unfortunately, the certainty of a response is not directly proportional magnitude. This occurs because the magnitude of even weak responses grow with time and because the components from cross-terms will cause responses to

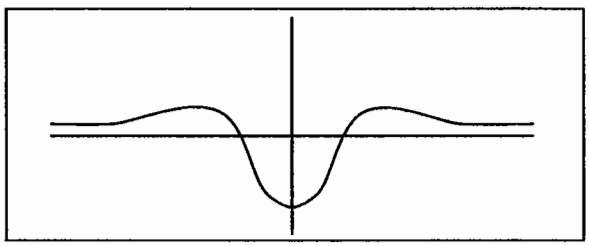

Figure 5: Lateral inhibition network connection strengths.

be longer than they should be. We claim that outupts that closely approximate valid effector encodings are most likely valid. Valid effector encodings are unimodal distributions with large peak to trough ratios. A reaction to a situation will only begin to assume this distribution if there has been consistent training with the similar actions with high utility. The degree to which an output approaches a valid encoding to determined with a non-linear lateral inhibition network.

4.2.2 Lateral Inhibition to Improve Learning Performance

We use a lateral inhibition network to post-process the outputs of the association matrix. In this lateral inhibition network each unit is connected to its nearest neighbors with slightly excitatory connections, to its next closest neighbors with slightly inhibitory connections and to its distant neighbors not at all. Figure 5 shows the strength of connections as a function of distance.

This type of network ensures that after a period of time only a small region of units will be strongly activated in any given area. To ensure that outputs remain bounded, the units employ a sigmoid output function[8]

Once post-processed by the lateral inhibition network the associator can no longer generate arbitrary outputs. All outputs will have only unit strongly activated. As a result, outputs are restricted to valid action encoding.

A lateral inhibition network can determine the certainty of an output of the associator. We claim certainty is inversely proportional to the time re-

[8]This functions maps the real line to values between 0 and 1.

quired for the post-processing network to stabilize[9]. When the input to the lateral inhibition network is unimodal and the standard deviation is small lateral inhibition takes a small number of iterations to stabilize. Conversely, inputs that have large standard deviation or multiple peaks will require more iterations to stabilize. While we are not aware of any mathematical characterizations for this form of lateral inhibition network our experimental evidence implies - and in some restricted cases Lippmann [5] has demonstrated - that this an accurate description of the stabilization process.

We define certainty as the inverse of stabilization time. As the robot moves sensor values will change; this causes the inputs to the lateral inhibition network to change. If at any time there is certain response to a situation we expect that the lateral inhibition network will stabilize which will in turn generate an effector command. Conversely, uncertain responses will not allow lateral inhibition to stabilize before the inputs change. This will preclude the generation of an effector command. Occasionally the network will generate spurious commands. We rely on the subsumption architecture to ignore them.

4.2.3 Alternative to implementing lateral inhibition

Lateral inhibition provides the theoretical functionality needed to improve the learning performance of our algorithm. However, in our actual implementation we use a much simpler and less computationally expensive technique. Within the separable subvectors, we look for a single spike that is at least some percentage larger than any of the other units. If the output passes this certainty test we can use its centroid to determine the output value. For our choice of encoding it appears that this simple processing is a reasonable approximation of lateral inhibition.

4.2.4 The Algorithm

The learning algorithm the robot uses is shown in figure 6.

5 Conclusions and Future Work

We believe the robot described in this paper demonstrates that:

[9]Stabilization is defined as the point where the units of the net stop changing significantly.

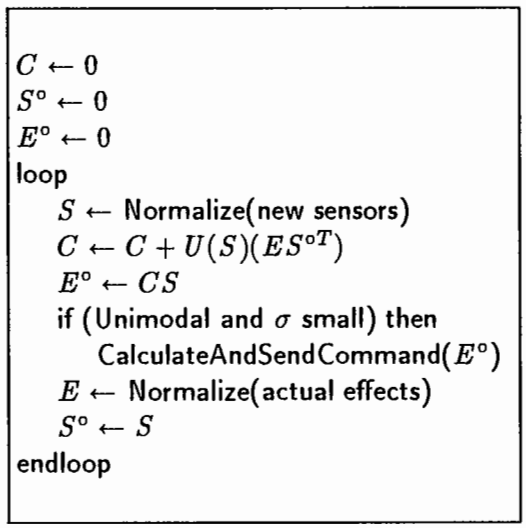

Figure 6: Simplied version of learning algorithm

- A subsumption architecture with derivative control and rapid cycle-time provides a good environment for a learning system.
- A very simple learning system can learn and direct a robot to perform novel and useful behaviors.
- Utility functions are a potent mechanism for directing learning.

We are currently attempting to use utility functions to guide formation of representations. It appears utility functions is one mechanism for learning deictic representations [1].

References

[1] P. E. Agre. The dynamic structure of everyday life. M.I.T. Ph.D. Thesis to be published as a TR, 1988.

[2] P. E. Agre and D. Chapmann. Pengi: An implementation of a theory of activity. *Proceedings of the Sixth National Conference on AI*, pages 196–201, 1987.

[3] R. A. Brooks and J. H. Connell. Asynchronous distributed control system for a mobile robot. *Cambridge Symposium on Optical and Optoelectronic Engineering*, pages 77–84, 1986.

[4] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23, 1986.

[5] Richard P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, April 1987.

[6] J.P. Shewchuk. An architecture for learning plans. Unpublished research notes. Brown University, Providence. RI.