We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

1

Robot Learning using Learning Classifier Systems Approach

Suraiya Jabin
Jamia Millia Islamia, Central University
(Department of Computer Science)
India

1. Introduction

Efforts to develop highly complex and adaptable machines that meet the ideal of mechanical human equivalents are now reaching the proof-of concept stage. Enabling a human to efficiently transfer knowledge and skills to a machine has inspired decades of research. I present a learning mechanism in which a robot learns new tasks using genetic-based machine learning technique, learning classifier system (LCS). LCSs are rule-based systems that automatically build their ruleset. At the origin of Holland's work, LCSs were seen as a model of the emergence of cognitive abilities thanks to adaptive mechanisms, particularly evolutionary processes. After a renewal of the field more focused on learning, LCSs are now considered as sequential decision problem-solving systems endowed with a generalization property. Indeed, from a Reinforcement Learning point of view, LCSs can be seen as learning systems building a compact representation of their problem. More recently, LCSs have proved efficient at solving automatic classification tasks (Sigaud et al., 2007). The aim of the present contribution is to describe the state-of the-art of LCSs, emphasizing recent developments, and focusing more on the application of LCS for Robotics domain.

In previous robot learning studies, optimization of parameters has been applied to acquire suitable behaviors in a real environment. Also in most of such studies, a model of human evaluation has been used for validation of learned behaviors. However, since it is very difficult to build human evaluation function and adjust parameters, a system hardly learns behavior intended by a human operator.

In order to reach that goal, I first present the two mechanisms on which they rely, namely GAs and Reinforcement Learning (RL). Then I provide a brief history of LCS research intended to highlight the emergence of three families of systems: strength-based LCSs, accuracy-based LCSs, and anticipatory LCSs (ALCSs) but mainly XCS as XCS, is the most studied LCS at this time. Afterward, in section 5, I present some examples of existing LCSs which have LCS applied for robotics. The next sections are dedicated to the particular aspects of theoretical and applied extensions of Intelligent Robotics. Finally, I try to highlight what seem to be the most promising lines of research given the current state of the art, and I conclude with the available resources that can be consulted in order to get a more detailed knowledge of these systems.

2. Basic formalism of LCS

A learning classifier system (LCS) is an adaptive system that learns to perform the best action given its input. By "best" is generally meant the action that will receive the most reward or reinforcement from the system's environment. By "input" is meant the environment as sensed by the system, usually a vector of numerical values. The set of available actions depends on the system context: if the system is a mobile robot, the available actions may be physical: "turn left", "turn right", etc. In a classification context, the available actions may be "yes", "no", or "benign", "malignant", etc. In a decision context, for instance a financial one, the actions might be "buy", "sell", etc. In general, an LCS is a simple model of an intelligent agent interacting with an environment.

A schematic depicting the rule and message system, the apportionment of credit system, and the genetic algorithm is shown in Figure 1. Information flows from the environment through the detectors-the classifier system's eyes and ears-where it is decoded to one or more finite length messages. These environmental messages are posted to a finite-length message list where the messages may then activate string rules called classifiers. When activated, a classifier posts a message to the message list. These messages may then invoke other classifiers or they may cause an action to be taken through the system's action triggers called effectors.

An LCS is "adaptive" in the sense that its ability to choose the best action improves with experience. The source of the improvement is reinforcement—technically, payoff provided by the environment. In many cases, the payoff is arranged by the experimenter or trainer of the LCS. For instance, in a classification context, the payoff may be 1.0 for "correct" and 0.0 for "incorrect". In a robotic context, the payoff could be a number representing the change in distance to a recharging source, with more desirable changes (getting closer) represented by larger positive numbers, etc. Often, systems can be set up so that effective reinforcement is provided automatically, for instance via a distance sensor. Payoff received for a given action

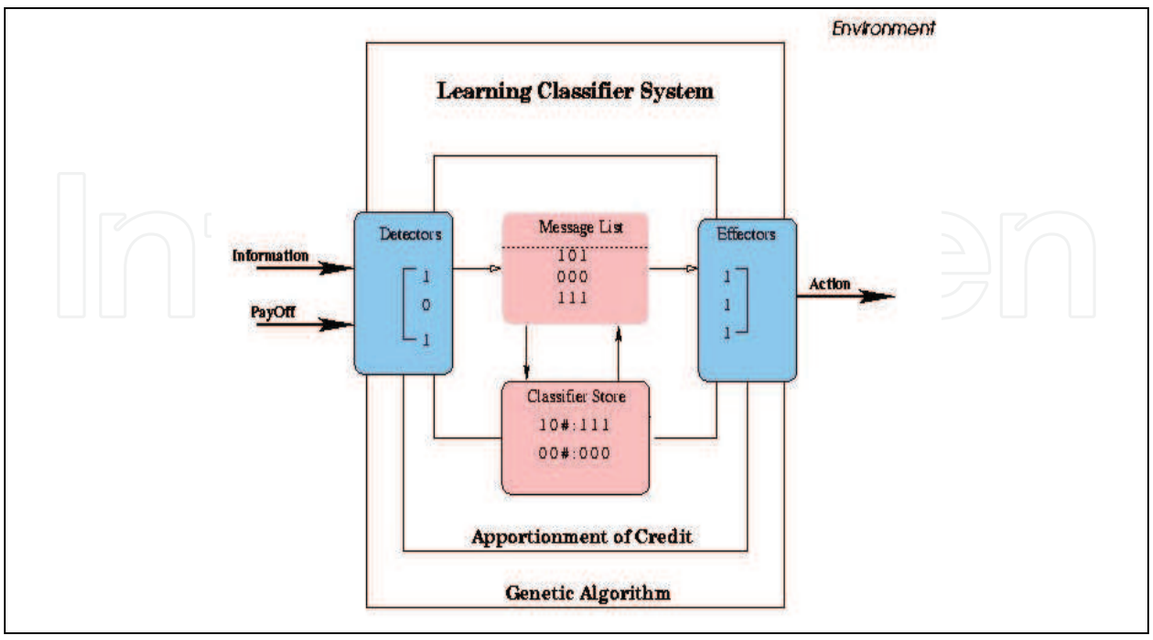


Fig. 1. A general Learning Classifier System

is used by the LCS to alter the likelihood of taking that action, in those circumstances, in the future. To understand how this works, it is necessary to describe some of the LCS mechanics.

Inside the LCS is a set technically, a population—of "condition-action rules" called classifiers. There may be hundreds of classifiers in the population. When a particular input occurs, the LCS forms a so-called match set of classifiers whose conditions are satisfied by that input. Technically, a condition is a truth function t(x) which is satisfied for certain input vectors x. For instance, in a certain classifier, it may be that t(x) = 1 (true) for $43 < x3 < 54$, where x3 is a component of x, and represents, say, the age of a medical patient. In general, a classifier's condition will refer to more than one of the input components, usually all of them. If a classifier's condition is satisfied, i.e. its t(x) = 1, then that classifier joins the match set and influences the system's action decision. In a sense, the match set consists of classifiers in the population that recognize the current input.

Among the classifiers—the condition-action rules—of the match set will be some that advocate one of the possible actions, some that advocate another of the actions, and so forth. Besides advocating an action, a classifier will also contain a prediction of the amount of payoff which, speaking loosely, "it thinks" will be received if the system takes that action. How can the LCS decide which action to take? Clearly, it should pick the action that is likely to receive the highest payoff, but with all the classifiers making (in general) different predictions, how can it decide? The technique adopted is to compute, for each action, an average of the predictions of the classifiers advocating that action—and then choose the action with the largest average. The prediction average is in fact weighted by another classifier quantity, its fitness, which will be described later but is intended to reflect the reliability of the classifier's prediction.

The LCS takes the action with the largest average prediction, and in response the environment returns some amount of payoff. If it is in a learning mode, the LCS will use this payoff, P, to alter the predictions of the responsible classifiers, namely those advocating the chosen action; they form what is called the action set. In this adjustment, each action set classifier's prediction p is changed mathematically to bring it slightly closer to P, with the aim of increasing its accuracy. Besides its prediction, each classifier maintains an estimate q of the error of its predictions. Like p, q is adjusted on each learning encounter with the environment by moving q slightly closer to the current absolute error $|p - P|$. Finally, a quantity called the classifier's fitness is adjusted by moving it closer to an inverse function of q, which can be regarded as measuring the accuracy of the classifier. The result of these adjustments will hopefully be to improve the classifier's prediction and to derive a measure—the fitness—that indicates its accuracy.

The adaptivity of the LCS is not, however, limited to adjusting classifier predictions. At a deeper level, the system treats the classifiers as an evolving population in which accurate i.e. high fitness—classifiers are reproduced over less accurate ones and the "offspring" are modified by genetic operators such as mutation and crossover. In this way, the population of classifiers gradually changes over time, that is, it adapts structurally. Evolution of the population is the key to high performance since the accuracy of predictions depends closely on the classifier conditions, which are changed by evolution.

Evolution takes place in the background as the system is interacting with its environment. Each time an action set is formed, there is finite chance that a genetic algorithm will occur in the set. Specifically, two classifiers are selected from the set with probabilities proportional

to their fitnesses. The two are copied and the copies (offspring) may, with certain probabilities, be mutated and recombined ("crossed"). Mutation means changing, slightly, some quantity or aspect of the classifier condition; the action may also be changed to one of the other actions. Crossover means exchanging parts of the two classifiers. Then the offspring are inserted into the population and two classifiers are deleted to keep the population at a constant size. The new classifiers, in effect, compete with their parents, which are still (with high probability) in the population.

The effect of classifier evolution is to modify their conditions so as to increase the overall prediction accuracy of the population. This occurs because fitness is based on accuracy. In addition, however, the evolution leads to an increase in what can be called the "accurate generality" of the population. That is, classifier conditions evolve to be as general as possible without sacrificing accuracy. Here, general means maximizing the number of input vectors that the condition matches. The increase in generality results in the population needing fewer distinct classifiers to cover all inputs, which means (if identical classifiers are merged) that populations are smaller and also that the knowledge contained in the population is more visible to humans—which is important in many applications. The specific mechanism by which generality increases is a major, if subtle, side-effect of the overall evolution.

3. Brief history of learning classifier systems

The first important evolution in the history of LCS research is correlated to the parallel progress in RL research, particularly with the publication of the Q-LEARNING algorithm (Watkins, 1989).

Classical RL algorithms such as Q-LEARNING rely on an explicit enumeration of all the states of the system. But, since they represent the state as a collection of a set of sensations called "attributes", LCSs do not need this explicit enumeration thanks to a generalization property that is described later. This generalization property has been recognized as the distinguishing feature of LCSs with respect to the classical RL framework. Indeed, it led Lanzi to define LCSs as RL systems endowed with a generalization capability (Lanzi, 2002).

An important step in this change of perspective was the analysis by Dorigo and Bersini of the similarity between the BUCKET BRIGADE algorithm (Holland, 1986) used so far in LCSs and the Q-LEARNING algorithm (Dorigo & Bersini, 1994). At the same time, Wilson published a radically simplified version of the initial LCS architecture, called Zeroth-level Classifier System ZCS (Wilson, 1994), in which the list of internal messages was removed.

ZCS defines the fitness or strength of a classifier as the accumulated reward that the agent can get from firing the classifier, giving rise to the "strength-based" family of LCSs. As a result, the GA eliminates classifiers providing less reward than others from the population.

After ZCS, Wilson invented a more subtle system called XCS (Wilson, 1995), in which the fitness is bound to the capacity of the classifier to accurately predict the reward received when firing it, while action selection still relies on the expected reward itself. XCS appeared very efficient and is the starting point of a new family of "accuracy-based" LCSs. Finally, two years later, Stolzmann proposed an anticipatory LCS called ACS (Stolzmann, 1998; Butz et al., 2000) giving rise to the "anticipation-based" LCS family.

This third family is quite distinct from the other two. Its scientific roots come from research in experimental psychology about latent learning (Tolman, 1932; Seward, 1949). More precisely, Stolzmann was a student of Hoffmann (Hoffmann, 1993) who built a

psychological theory of learning called "Anticipatory Behavioral Control" inspired from Herbart's work (Herbart, 1825).
The extension of these three families is at the heart of modern LCS research. Before closing this historical overview, after a second survey of the field (Lanzi and Riolo, 2000), a further important evolution is taking place. Even if the initial impulse in modern LCS research was based on the solution of sequential decision problems, the excellent results of XCS on data mining problems (Bernado et al., 2001) have given rise to an important extension of researches towards automatic classification problems, as exemplified by Booker (2000) or Holmes (2002).

4. Mechanisms of learning classifier systems

4.1 Genetic algorithm

First, I briefly present GAs (Holland, 1975; Booker et al., 1989; Goldberg, 1989), which are freely inspired from the neo-darwinist theory of natural selection. These algorithms manipulate a population of individuals representing possible solutions to a given problem. GAs rely on four analogies with their biological counterpart: they use a code, the genotype or genome, simple transformations operating on that code, the genetic operators, the expression of a solution from the code, the genotype-to-phenotype mapping, and a solution selection process, the survival of the fittest. The genetic operators are used to introduce some variations in the genotypes. There are two classes of operators: crossover operators, which create new genotypes by recombining sub-parts of the genotypes of two or more individuals, and mutation operators, which randomly modify the genotype of an individual. The selection process extracts the genotypes that deserve to be reproduced, upon which genetic operators will be applied. A GA manipulates a set of arbitrarily initialized genotypes which are selected and modified generation after generation. Those which are not selected are eliminated. A utility function, or fitness function, evaluates the interest of a phenotype with regard to a given problem. The survival of the corresponding solution or its number of offspring in the next generation depends on this evaluation. The offspring of an individual are built from copies of its genotype to which genetic operators are applied. As a result, the overall process consists in the iteration of the following loop:

1. select n_e genotypes according to the fitness of corresponding phenotypes,
2. apply genetic operators to these genotypes to generate offspring,
3. build phenotypes from these new genotypes and evaluate them,
4. go to 1.

If some empirical conditions that we will not detail here are fulfilled, such a process gives rise to an improvement of the fitnesses of the individuals over the generations.
Though GAs are at their root, LCSs have made limited use of the important extensions of this field. As a consequence, in order to introduce the GAs used in LCSs, it is only necessary to describe the following aspects:

a. One must classically distinguish between the one-point crossover operator, which cuts two genotypes into two parts at a randomly selected place and builds a new genotype by inverting the sub-parts from distinct parents, and the multi-point crossover operator, which does the same after cutting the parent genotypes into several pieces. Historically, most early LCSs were using the one-point crossover operator. Recently, a surge of interest on the discovery of complex 'building blocks' in the structure of input data led to a more frequent use of multi-point crossover.

b. One must also distinguish between generational GAs, where all or an important part of the population is renewed from one generation to the next, and steady state GAs, where individuals are changed in the population one by one without notion of generation. Most LCSs use a steady-state GA, since this less disruptive mechanism results in a better interplay between the evolutionary process and the learning process, as explained below.

4.2 Markov Decision Processes and reinforcement learning

The second fundamental mechanism in LCSs is Reinforcement Learning. In order to describe this mechanism, it is necessary to briefly present the Markov Decision Process (MDP) framework and the Q-LEARNING algorithm, which is now the learning algorithm most used in LCSs. This presentation is as succinct as possible; the reader who wants to get a deeper view is referred to Sutton and Barto (1998).

4.2.1 Markov Decision Processes

A MDP is defined as the collection of the following elements:

- a finite set S of discrete states s of an agent;
- a finite set A of discrete actions a;
- a transition function P : S X A → ∏ (S) where ∏ (S) is the set of probability distributions over S. A particular probability distribution Pr(s_{t+1} | st, a_t) indicates the probabilities that the agent reaches the different s_{t+1} possible states when he performs action a_t in state s_t;
- a reward function R : S ×A → IR which gives for each (st, at) pair the scalar reward signal that the agent receives when he performs action a_t in state s_t .

The MDP formalism describes the stochastic structure of a problem faced by an agent, and does not tell anything about the behavior of this agent in its environment. It only tells what, depending on its current state and action, will be its future situation and reward.

The above definition of the transition function implies a specific assumption about the nature of the state of the agent. This assumption, known as the Markov property, stipulates that the probability distribution specifying the s_{t+1} state only depends on s_t and a_t, but not on the past of the agent. Thus P(s_{t+1} | s_t, a_t) = P(s_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, . . . , s_0, a_0). This means that, when the Markov property holds, a knowledge of the past of the agent does not bring any further information on its next state.

The behavior of the agent is described by a policy ∏ giving for each state the probability distribution of the choice of all possible actions.

When the transition and reward functions are known in advance, Dynamic Programming (DP) methods such as policy iteration (Bellman, 1961; Puterman & Shin, 1978) and value iteration (Bellman, 1957) efficiently find a policy maximizing the accumulated reward that the agent can get out of its behavior.

In order to define the accumulated reward, we introduce the discount factor $\gamma \in [0, 1]$. This factor defines how much the future rewards are taken into account in the computation of the accumulated reward at time *t* as follows:

$$Rc_{\pi}(t) = \sum_{k=t}^{T_{\max}} \gamma^{(k-t)} r_{\pi}(k)$$

where T_{max} can be finite or infinite and $r_\pi(k)$ represents the immediate reward received at time k if the agent follows policy π.
DP methods introduce a value function V^π where $V^\pi(s)$ represents for each state s the accumulated reward that the agent can expect if it follows policy π from state s. If the Markov property holds, V^π is solution of the Bellman equation (Bertsekas, 1995):

$$\forall s \in S, V^\pi(s) = \sum_a \pi(s_t, a_t)[R(s_t, a_t) + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V^\pi(s_{t+1})] \quad (1)$$

Rather than the value function V^π, it is often useful to introduce an action value function Q^π where $Q^\pi(s, a)$ represents the accumulated reward that the agent can expect if it follows policy π after having done action a in state s. Everything that was said of V^π directly applies to Q^π, given that $V^\pi(s) = max_a Q^\pi(s, a)$.
The corresponding optimal functions are independent of the policy of the agent; they are denoted V* and Q*.
(a) The manuscript must be written in English, (b) use common technical terms, (c) avoid

4.2.2 Reinforcement learning

Learning becomes necessary when the transition and reward functions are not known in advance. In such a case, the agent must explore the outcome of each action in each situation, looking for the (s_t, a_t) pairs that bring it a high reward.
The main RL methods consist in trying to estimate V* or Q* iteratively from the trials of the agent in its environment. All these methods rely on a general approximation technique in order to estimate the average of a stochastic signal received at each time step without storing any information from the past of the agent. Let us consider the case of the average immediate reward. Its exact value after k iterations is

$E_k(s) = (r_1 + r_2 + \cdots + r_k)/k$

Furthermore,

$E_{k+1}(s) = (r_1 + r_2 + \cdots + r_k + r_{k+1})/(k + 1)$

thus

$E_{k+1}(s) = k/(k + 1)\, E_k(s) + r_{k+1}/(k + 1)$

which can be rewritten:

$E_{k+1}(s) = (k + 1)/(k + 1)\, E_k(s) - E_k(s)/(k + 1) + r_{k+1}/(k + 1)$

or

$E_{k+1}(s) = E_k(s) + 1/(k + 1)[r_{k+1} - E_k(s)]$

Formulated that way, we can compute the exact average by merely storing k. If we do not want to store even k, we can approximate $1/(k + 1)$ with α, which results in equation (2) whose general form is found everywhere in RL:

$$E_{k+1}(s) = E_k(s) + \alpha[r_{k+1} - E_k(s)] \quad (2)$$

The parameter α, called learning rate, must be tuned adequately because it influences the speed of convergence towards the exact average.
The update equation of the Q-LEARNING algorithm, which is the following:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (3)$$

5. Some existing LCSs for robotics

LCSs were invented by Holland (Holland, 1975) in order to model the emergence of cognition based on adaptive mechanisms. They consist of a set of rules called classifiers combined with adaptive mechanisms in charge of evolving the population of rules. The initial goal was to solve problems of interaction with an environment such as the one presented in figure 2, as was described by Wilson as the "Animat problem" (Wilson, 1985).
In the context of the initial research on LCSs, the emphasis was put on parallelism in the architecture and evolutionary processes that let it adapt at any time to the variations of the environment (Golberg & Holland, 1988). This approach was seen as a way of "escaping brittleness" (Holland, 1986) in reference to the lack of robustness of traditional artificial intelligence systems faced with problems more complex than toy or closed-world problems.

5.1 Pittsburgh versus Michigan

This period of research on LCSs was structured by the controversy between the so-called "Pittsburgh" and "Michigan" approaches. In Smith's approach (Smith, 1980), from the University of Pittsburgh, the only adaptive process was a GA applied to a population of LCSs in order to choose from among this population the fittest LCS for a given problem.
By contrast, in the systems from Holland and his PhD students, at the University of Michigan, the GA was combined since the very beginning with an RL mechanism and was applied more subtly within a single LCS, the population being represented by the set of classifiers in this system.
Though the Pittsburgh approach is becoming more popular again currently, (Llora & Garrell, 2002; Bacardit & Garrell, 2003; Landau et al., 2005), the Michigan approach quickly became the standard LCS framework, the Pittsburgh approach becoming absorbed into the wider evolutionary computation research domain.

5.2 The ANIMAT classifier system

Inspired by Booker's two-dimensional critter, Wilson developed a roaming classifier system that searched a two-dimensional jungle, seeking food and avoiding trees. Laid out on an 18 by 58 rectangular grid, each woods contained clusters of trees (T's) and food (F's) placed in regular clusters about the space. A typical woods is shown in figure 2. The ANIMAT (represented by a *) in a woods has knowledge concerning his immediate surroundings. For example, ANIMAT is surrounded by two trees (T), one food parcel (F), and blank spaces (B) as shown below:

B T T

B * F

B B B

This pattern generates an environmental message by unwrapping a string starting at compass north and moving clockwise:

T T F B B B B B

Under the mapping T→01, F→11, B→00 (the first position may be thought of as a binary smell detector and the second position as a binary opacity detector) the following message is generated:

0101110000000000

ANIMAT responds to environmental messages using simple classifiers with 16-position condition (corresponding to the 16-position message) and eight actions (actions 0-7). Each action corresponds to a one-step move in one of the eight directions (north, north east, east and so on).

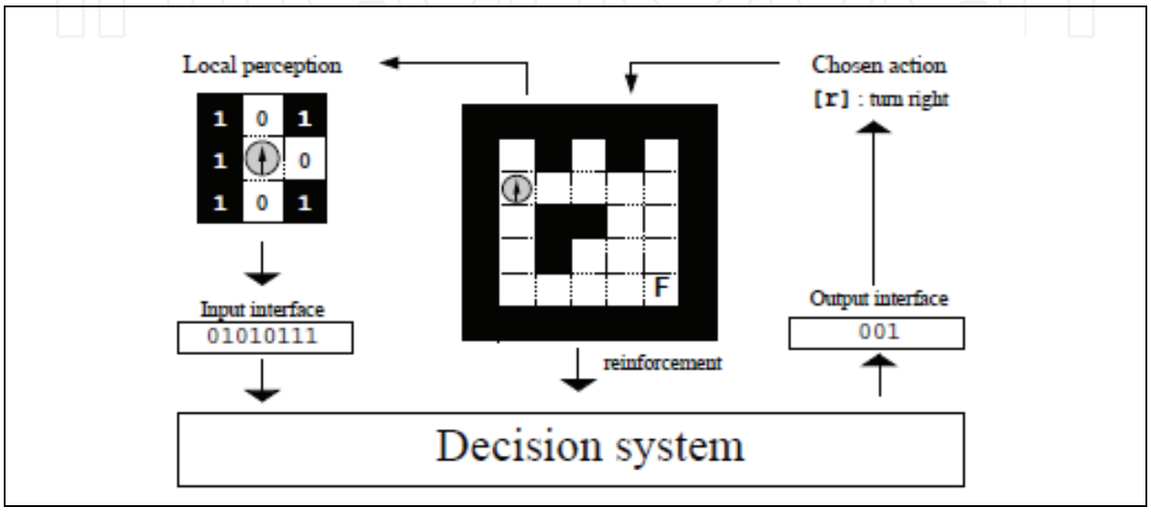


Fig. 2. Representation of an interaction problem. The agent senses a situation as a set of attributes. In this example, it is situated in a maze and senses either the presence (symbol 1) or the absence (symbol 0) of walls in the eight surrounding cells, considered clockwise starting from the north. Thus, in the above example it senses [01010111]. This information is sent to its input interface. At each time step, the agent must choose between going forward [f], turning right [r] or left [l]. The chosen action is sent through the output interface.

It is remarkable that ANIMAT learned the task as well as it did considering how little knowledge it actually possessed. For it to do much better, it would have to construct a mental map of the woods so it could know where to go when it was surrounded by blanks. This kind of internal modelling can be developed within a classifier system framework; however work in this direction has been largely theoretical.

5.3 Interactive classifier system for real robot learning

Reinforcement learning has been applied to robot learning in a real environment (Uchibe et al., 1996). In contrast with modeling human evaluation analytically, another approach is introduced in which a system learns suitable behavior using human direct evaluation without its modeling. Such an interactive method with *Evolutionary Computation (EC)* as a search algorithm is called *Interactive* EC (Dawkins, 1989), and a lot of studies on it have been done thus far (Nakanishi; Oshaki et al.; Unemi). The most significant issue of *Interactive EC* is how it reduces human teaching load. The human operator needs to evaluate a lot of individuals at every generation, and this evaluation makes him/her so tired. Specially in the

interactive EC applied to robotics, the execution of behaviors by a robot significantly costs and a human operator can not endure such a boring task. Additionally reinforcement learning has been applied to robot learning in a real environment (Uchibe et al., 1996). Unfortunately the learning takes pretty much time to converge. Furthermore, when a robot hardly gets the first reward because of no priori knowledge, the learning convergence becomes far slower. Since most of the time that are necessary for one time of action moreover is spent in processing time of sense system and action system of a robot, the reduction of learning trials is necessary to speedup the learning.
In the Interactive Classifier System (D. Katagami et al., 2000), a human operator instructs a mobile robot while watching the information that a robot can acquire as sensor information and camera information of a robot shown on the screen top. In other words, the operator acquires information from a viewpoint of a robot instead of a viewpoint of a designer. In this example, an *interactive **EC*** framework is build which quickly learns rules with operation signal of a robot by a human operator as teacher signal. Its objective is to make initial learning more efficient and learn the behaviors that a human operator intended through interaction with him/her. To the purpose, a classifier system is utilized as a learner because it is able to learn suitable behaviors by the small number of trials, and also extend the classifier system to be adaptive to a dynamic environment.
In this system, a human operator instructs a mobile robot while watching the information that a robot can acquire as sensor information and camera information of a robot shown on the screen top. In other words, the operator acquires information from a viewpoint of a robot instead of a viewpoint of a designer. Operator performs teaching with joystick by direct operating a physical robot. The ICS inform operator about robot's state by a robot send a vibration signal of joystick to the ICS according to inside state. This system is a fast learning method based on ICS for mobile robots which acquire autonomous behaviors from experience of interaction between a human and a robot.

6. Intelligent robotics: past, present and future

Robotics began in the 1960s as a field studying a new type of universal machine implemented with a computer-controlled mechanism. This period represented an age of over expectation, which inevitably led to frustration and discontent with what could realistically be achieved given the technological capabilities at that time. In the 1980s, the field entered an era of realism as engineers grappled with these limitations and reconciled them with earlier expectations. Only in the past few years have we achieved a state in which we can feasibly implement many of those early expectations. As we do so, we enter the 'age of exploitation (Hall, 2001).
For more than 25 years, progress in concepts and applications of robots have been described, discussed, and debated. Most recently we saw the development of 'intelligent' robots, or robots designed and programmed to perform intricate, complex tasks that require the use of adaptive sensors. Before we describe some of these adaptations, we ought to admit that some confusion exists about what intelligent robots are and what they can do. This uncertainty traces back to those early over expectations, when our ideas about robots were fostered by science fiction or by our reflections in the mirror. We owe much to their influence on the field of robotics. After all, it is no coincidence that the submarines or airplanes described by Jules Verne and Leonardo da Vinci now exist. Our ideas have origins,

and the imaginations of fiction writers always ignite the minds of scientists young and old, continually inspiring invention. This, in turn, inspires exploitation.
We use this term in a positive manner, referring to the act of maximizing the number of applications for, and usefulness of inventions.
Years of patient and realistic development have tempered our definition of intelligent robots. We now view them as mechanisms that may or may not look like us but can perform tasks as well as or better than humans, in that they sense and adapt to changing requirements in their environments or related to their tasks, or both. Robotics as a science has advanced from building robots that solve relatively simple problems, such as those presented by games, to machines that can solve sophisticated problems, like navigating dangerous or unexplored territory, or assisting surgeons. One such intelligent robot is the autonomous vehicle. This type of modern, sensor-guided, mobile robot is a remarkable combination of mechanisms, sensors, computer controls, and power sources, as represented by the conceptual framework in Figure 3. Each component, as well as the proper interfaces between them, is essential to building an intelligent robot that can successfully perform assigned tasks.

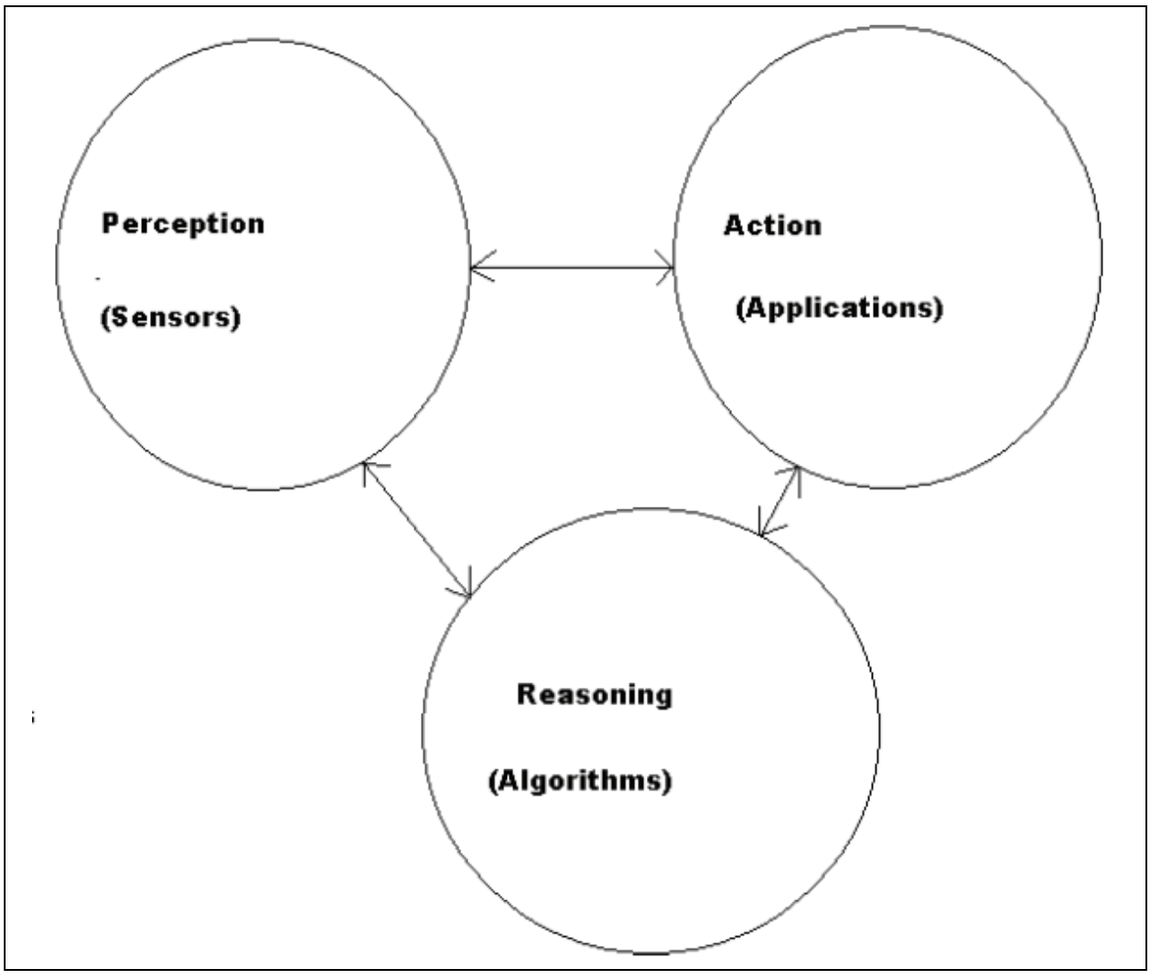


Fig. 3. Conceptual framework of components for intelligent robot design.

An example of an autonomous-vehicle effort is the work of the University of Cincinnati Robot Team. They exploit the lessons learned from several successive years of autonomous ground-vehicle research to design and build a variety of smart vehicles for unmanned operation. They have demonstrated their robots for the past few years (see Figure 2) at the Intelligent Ground Vehicle Contest and the Defense Advanced Research Project Agency's (DARPA) Urban Challenge.

Fig. 4. 'Bearcat Cub' intelligent vehicle designed for the Intelligent Ground Vehicle Contest

These and other intelligent robots developed in recent years can look deceptively ordinary and simple. Their appearances belie the incredible array of new technologies and methodologies that simply were not available more than a few years ago. For example, the vehicle shown in Figure 4 incorporates some of these emergent capabilities. Its operation is based on the theory of dynamic programming and optimal control defined by Bertsekas,5 and it uses a problem-solving approach called backwards induction. Dynamic programming permits sequential optimization. This optimization is applicable to mechanisms operating in nonlinear, stochastic environments, which exist naturally.

It requires efficient approximation methods to overcome the high-dimensionality demands. Only since the invention of artificial neural networks and backpropagation has this powerful and universal approach become realizable. Another concept that was incorporated into the robot is an eclectic controller (Hall et al., 2007). The robot uses a real-time controller to orchestrate the information gathered from sensors in a dynamic environment to perform tasks as required. This eclectic controller is one of the latest attempts to simplify the operation of intelligent machines in general, and of intelligent robots in particular. The idea is to use a task-control center and dynamic programming approach with learning to optimize performance against multiple criteria.

Universities and other research laboratories have long been dedicated to building autonomous mobile robots and showcasing their results at conferences. Alternative forums for exhibiting advances in mobile robots are the various industry or government sponsored competitions. Robot contests showcase the achievements of current and future roboticists and often result in lasting friendships among the contestants. The contests range from those for students at the highest educational level, such as the DARPA Urban Challenge, to K-12 pupils, such as the First Lego League and Junior Lego League Robotics competitions. These contests encourage students to engage with science, technology, engineering, and mathematics, foster critical thinking, promote creative problem solving, and build

professionalism and teamwork. They also offer an alternative to physical sports and reward scholastic achievement.

Why are these contests important, and why do we mention them here? Such competitions have a simple requirement, which the entry either works or does not work. This type of proof-of concept pervades many creative fields. Whether inventors showcase their work at conferences or contests, most hope to eventually capitalize on and exploit their inventions, or at least appeal to those who are looking for new ideas, products, and applications.

As we enter the age of exploitation for robotics, we can expect to see many more proofs-of-concept following the advances that have been made in optics, sensors, mechanics, and computing. We will see new systems designed and existing systems redesigned. The challenges for tomorrow are to implement and exploit the new capabilities offered by emergent technologies—such as petacomputing and neural networks—to solve real problems in real time and in cost-effective ways. As scientists and engineers master the component technologies, many more solutions to practical problems will emerge. This is an exciting time for roboticists. We are approaching the ability to control a robot that is becoming as complicated in some ways as the human body. What could be accomplished by such machines? Will the design of intelligent robots be biologically inspired or will it continue to follow a completely different framework? Can we achieve the realization of a mathematical theory that gives us a functional model of the human brain, or can we develop the mathematics needed to model and predict behavior in large scale, distributed systems? These are our personal challenges, but all efforts in robotics—from K-12 students to established research laboratories—show the spirit of research to achieve the ultimate in intelligent machines. For now, it is clear that roboticists have laid the foundation to develop practical, realizable, intelligent robots. We only need the confidence and capital to take them to the next level for the benefit of humanity.

7. Conclusion

In this chapter, I have presented Learning Classifier Systems, which add to the classical Reinforcement Learning framework the possibility of representing the state as a vector of attributes and finding a compact expression of the representation so induced. Their formalism conveys a nice interaction between learning and evolution, which makes them a class of particularly rich systems, at the intersection of several research domains. As a result, they profit from the accumulated extensions of these domains.

I hope that this presentation has given to the interested reader an appropriate starting point to investigate the different streams of research that underlie the rapid evolution of LCS. In particular, a key starting point is the website dedicated to the LCS community, which can be found at the following URL: http://lcsweb.cs.bath.ac.uk/.

8. References

Bacardit, J. and Garrell, J. M. (2003). Evolving multiple discretizations with adaptive intervals for a Pittsburgh rule-based learning classifier system. In Cantú Paz, E., Foster, J. A., Deb, K., Davis, D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A.,

Schultz, A. C., Dowsland, K., Jonoska, N., and Miller, J., (Eds.), Genetic and Evolutionary Computation – GECCO-2003, pages 1818–1831, Berlin. Springer-Verlag.

Bellman, R. E. (1957). Dynamic Programming. Princeton University Press, Princeton, NJ.

Bellman, R. E. (1961). Adaptive Control Processes: A Guided Tour. Princeton University Press.

Bernado, E., Llorá, X., and Garrel, J. M. (2001). XCS and GALE : a comparative study of two Learning Classifer Systems with six other learning algorithms on classification tasks. In Lanzi, P.-L., Stolzmann, W., and Wilson, S. W., (Eds.), Proceedings of the fourth international workshop on Learning Classifer Systems.

Booker, L., Goldberg, D. E., and Holland, J. H. (1989). Classifier Systems and Genetic Algorithms. Artificial Intelligence, 40(1-3):235–282.

Booker, L. B. (2000). Do we really need to estimate rule utilities in classifier systems? In Lanzi, P.-L., Stolzmann, W., and Wilson, S. W., (Eds.), Learning Classifier Systems. From Foundations to Applications, volume 1813 of Lecture Notes in Artificial Intelligence, pages 125–142, Berlin. Springer-Verlag.

Dorigo, M. and Bersini, H. (1994). A comparison of Q-Learning and Classifier Systems. In Cliff, D., Husbands, P., Meyer, J.-A., and Wilson, S. W., (Eds.), From Animals to Animats 3, pages 248–255, Cambridge, MA. MIT Press.

Golberg, D. E. and Holland, J. H. (1988). Guest Editorial: Genetic Algorithms and Machine Learning. Machine Learning, 3:95–99.

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley, Reading, MA.

Hall, E. L. (2001), Intelligent robot trends and predictions for the .net future, Proc. SPIE 4572, pp. 70–80, 2001. doi:10.1117/12.444228

Hall, E. L., Ghaffari M., Liao X., Ali S. M. Alhaj, Sarkar S., Reynolds S., and Mathur K., (2007).*Eclectic theory of intelligent robots*, Proc. SPIE 6764, p. 676403, 2007. doi:10.1117/12.730799

Herbart, J. F. (1825). Psychologie als Wissenschaft neu gegr¨undet auf Erfahrung, Metaphysik und Mathematik. Zweiter, analytischer Teil. AugustWilhem Unzer, Koenigsberg, Germany.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press, Ann Arbor, MI.

Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Machine Learning, An Artificial Intelligence Approach (volume II). Morgan Kaufmann.

Holmes, J. H. (2002). A new representation for assessing classifier performance in mining large databases. In Stolzmann, W., Lanzi, P.-L., and Wilson, S. W., (Eds.), IWLCS-02. Proceedings of the International Workshop on Learning Classifier Systems, LNAI, Granada. Springer-Verlag.

Katagami, D.; Yamada, S. (2000). Interactive Classifier System for Real Robot Learning, *Proceedings of the 2000 IEEE International Workshop on Robot and Human Interactive Communication,* pp. 258-264, ISBN 0-7803-6273, Osaka, Japan, September 27-29 2000

Landau, S., Sigaud, O., and Schoenauer, M. (2005). ATNoSFERES revisited. In Beyer, H.-G., O'Reilly, U.-M., Arnold, D., Banzhaf, W., Blum, C., Bonabeau, E., Cant Paz, E., Dasgupta, D., Deb, K., Foste r, J., de Jong, E., Lipson, H., Llora, X., Mancoridis, S., Pelikan, M., Raidl, G., Soule, T., Tyrrell, A., Watson, J.-P., and Zitzler, E., (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005, pages 1867–1874, Washington DC. ACM Press.

Lanzi, P.-L. (2002). Learning Classifier Systems from a Reinforcement Learning Perspective. Journal of Soft Computing, 6(3-4):162–170.

Ohsaki, M., Takagi H. and T. Ingu. Methods to Reduce the Human Burden of Interactive Evolutionary Computation. Asian Fuzzy System Symposium (AFSS'98), pages 4955500, 1998.

Puterman, M. L. and Shin, M. C. (1978). Modified Policy Iteration Algorithms for Discounted Markov Decision Problems. Management Science, 24:1127–1137.

R. Dawkins. *TlLe Blind Watchmaker*. Longman, Essex, 1986.

R. Dawkins. The Evolution of Evolvability. In Langton, C. G., editor, Artificial Life, pages 201-220. Addison-Wesley, 1989.

Seward, J. P. (1949). An Experimental Analysis of Latent Learning. Journal of Experimental Psychology, 39:177–186.

Sigaud, O. and Wilson, S.W. (2007). Learning Classifier Systems: A Survey, Journal of Soft Computing, Springer-Verlag (2007)

Smith, S. F. (1980). A Learning System Based on Genetic Algorithms. PhD thesis, Department of Computer Science, University of Pittsburg, Pittsburg, MA.

Stolzmann, W. (1998). Anticipatory Classifier Systems. In Koza, J., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., and Riolo, R., (Eds.), Genetic Programming, pages 658–664. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

Sutton, R. S. and Barto, A. G. (1998). Reinforcement Learning: An Introduction. MIT Press.

Tolman, E. C. (1932). Purposive behavior in animals and men. Appletown, New York.

Uchibe, E., Asad M. and Hosoda, K. Behavior coordination for a mobile robot using modular reinforcement learning. In IEEE/RSJ International Conference on Intelligent Robots and Systems 1996 (IROS96), pages 1329-1336, 1996.

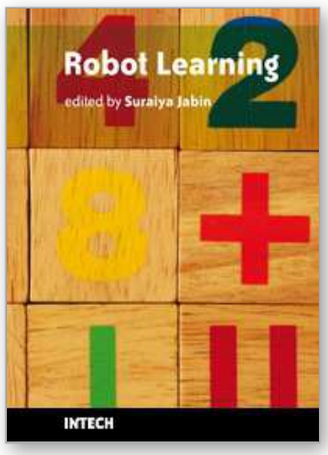
Wilson, S. W. (1985). Knowledge Growth in an Artificial Animat. In Grefenstette, J. J., (Ed.), Proceedings of the 1st international Conference on Genetic Algorithms and their applications (ICGA85), pages 16–23. L. E. Associates.

Wilson, S. W. (1994). ZCS, a Zeroth level Classifier System. Evolutionary Computation, 2(1):1–18.

Wilson, S. W. (1995). Classifier Fitness Based on Accuracy. Evolutionary Computation, 3(2):149–175.

Y. Nakanishi. Capturing Preference into a Function Using Interactions with a Manual Evolutionary Design Aid System. Genetic Programming, pages 133-140, 1996.

University of Cincinnati robot team. http://www.robotics.uc.edu
Intelligent Ground Vehicle Contest. http://www.igvc.org
Defense Advanced Research Project Agency's Urban Challenge. http: //www. darpa. mil / grandchallenge



Robot Learning

Edited by Suraiya Jabin

ISBN 978-953-307-104-6

Hard cover, 150 pages

Publisher Sciyo

Published online 12, August, 2010

Published in print edition August, 2010

Robot Learning is intended for one term advanced Machine Learning courses taken by students from different computer science research disciplines. This text has all the features of a renowned best selling text. It gives a focused introduction to the primary themes in a Robot learning course and demonstrates the relevance and practicality of various Machine Learning algorithms to a wide variety of real-world applications from evolutionary techniques to reinforcement learning, classification, control, uncertainty and many other important fields. Salient features: - Comprehensive coverage of Evolutionary Techniques, Reinforcement Learning and Uncertainty. - Precise mathematical language used without excessive formalism and abstraction. - Included applications demonstrate the utility of the subject in terms of real-world problems. - A separate chapter on Anticipatory-mechanisms-of-human-sensory-motor-coordination and biped locomotion. - Collection of most recent research on Robot Learning.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen