Working Paper 93-18
Statistics and Econometrics Series 14
September 1993

Departamento de Estadística y Econometría
Universidad Carlos III de Madrid
Calle Madrid, 126
28903 Getafe (Spain)
Fax (341) 624-9849

INFERENCE IN CLASSIFIER SYSTEMS

Jorge Muruzábal*

Abstract

Classifier systems (Css) provide a rich framework for learning and induction, and they have been successfully applied in the artificial intelligence literature for some time. In this paper, both the architecture and the inferential mechanisms in general CSs are reviewed, and a number of limitations and extensions of the basic approach are summarized. A system based on the CS approach that is capable of quantitative data analysis is outlined and some of its peculiarities discussed.

Key Words
Classifier systems; Machine learning; Exploratory data analysis.

*Departamento de Estadística y Econometría, Universidad Carlos III de Madrid.

Inference in classifier systems

Jorge Muruzábal[1]
Departamento de Estadística y Econometría
Universidad Carlos III de Madrid

Abstract

Classifier systems (Css) provide a rich framework for learning and induction, and they have been successfully applied in the artificial intelligence literature for some time. In this paper, both the architecture and the inferential mechanisms in general CSs are reviewed, and a number of limitations and extensions of the basic approach are summarized. A system based on the CS approach that is capable of quantitative data analysis is outlined and some of its peculiarities discussed.

Keywords: Classifier systems; Machine learning; Exploratory data analysis.

1 Introduction

Classifier systems (Css) provide a rich framework for learning and induction, and they have been successfully applied in the artificial intelligence literature for some time. John Holland is the creator of the research area often called *genetic learning*, the branch of machine learning centered around the idea of artificial adaptation. Genetic learning essentially encompasses the theory of genetic algorithms (GAs, introduced in [Holland 1975]) and CSs. Although early CS-like ideas can be traced back to [Holland 1975], the system proposed by Holland and Reitman [1978] is usually credited as the first CS. The original approach has since evolved towards what could be called the standard model (described in detail in [Holland 1985,1986a,1989]). A comprehensive account is provided in [Holland et al. 1986].

Throughout the last decade, the basic ideas have been notably enhanced in various directions. First, empirical data abound today; many implementations have been shown to learn to solve problems of varying nature and difficulty, and useful insights on the internal dynamics are available. Second, mathematical frameworks allowing the study of analytical properties have been put forward, [Holland 1986b], [Arthur 1990]. Third, CSs have been shown to provide a successful framework for the *emergence* of structure, [Riolo 1989ab,1991]. A knowledge structure is said to emerge when the system is not told explicitly how to build it. Emergent computation mimics in a deep and more realistic way the learning process: since it let us proceed without compromising a few extra degrees of freedom, it probably give us a better chance to understand complex phenomena, see [Forrest 1990], [Lane 1992]. Finally, the main framework has also diversified substantially: certain peculiarities and difficulties in the standard model have often given place to a number of alternatives and extensions thereof.

---

[1]This article is based on the author's Ph. D. dissertation. Support for this research has been provided in part by grant NSF/DMS-8911548-02 (U.S.A.), by the Economics Research Program at the Santa Fe Institute, Santa Fe, NM, and by grant DGICYT PB92-0246 (Spain).

In this paper, we first briefly review the the standard CS model (section 2), then summarize some extensions (section 3). Finally, section 4 sketches a simple CS called PASS (Predictive Adaptive Sequential System, [Muruzábal 1992,1993]) capable of carrying out certain type of quantitative data analysis.

2 The basic model

A classifier system (CS) models the interaction between a learner and an environment where the learner is inmersed. The learner consists of performance and learning systems. The performance system takes care of maintaining a dialogue with the environment; the learning system oversees the performance system and introduces suitable modifications leading to better performance. The nucleus of the performance system is a collection of production rules called classifiers. These classifiers process the information perceived from the environment and encode current beliefs. The performance system is also endowed with a global memory buffer called the message list. The message list may contain previously observed events or the learner's preferences (desires) at a given point. The combination of message list and environmental input guides the system's actions as determined by the current population of classifiers.

CS syntax involves messages and schemata. Messages are strings in the n-dimensional boolean hypercube. Schemata are subsets of this space formed by specifying a number of common coordinates. Environmental input is represented as the sum of one nonnegative scalar called reinforcement R and one message m_E; the message list contains a fixed, prespecified number of messages. Both the scalar and all the previous messages change every time step.

Classifiers are typically defined as structures (i_1,i_2,o,S), where i_1, i_2 and o are schemata and S is a nonnegative scalar called strength. The information contained in a given classifier is interpreted in general as follows: If the environmental message m_E matches i_1 and some message m_L in the message list (or perhaps m_E again) matches i_2, then propose to output a message $m_O=o(m_E,m_L)$ with strength-dependent intensity. The output message can be either posted in the message list for the next time step or induce a specific action to be carried out by the system or both. In turn, this action may influence the next environmental input, which closes the interaction cycle. Whether a particular message is actually activated (and the action effected) depends on competing proposals. The competition process is randomized and influenced by strength and other features of the overall situation (see 3.2).

The amount of reinforcement R entering the system guides the system's learning mechanisms in two ways. First, relatively large R is provided when the system's actions lead to certain target states. Classifiers directly responsible for those successful actions typically increase their strength, while classifiers that either do not participate in the decision-making process or lead to scant R typically decrease their strength. Since decisions are based on strength, successful classifiers are thus more and more likely to be used in similar future situations.

Second, new classifiers are injected by the system's discovery mechanisms. These usually replace low-strength classifiers with new classifiers obtained from the strongest available so far. This search process is usually randomized and based on loose inductive biases acting nearly always at the syntactic level. The GA is expected to be useful and many CSs include it, but results have not been very promising and many additional mechanisms

have been devised (see 3.1).

Since classifiers are relatively primitive epistomological units, it is crucial to notice that much can be accomplished by combining classifiers into structures. Most often structure is expected to emerge on its own, although sometimes a bit of structure is built into the CS from the start, see eg. [Shu and Schaeffer 1991]. The power of the CS approach as general knowledge representation tool in a nonlearning context is shown eg. in [Forrest 1991]. Let us discuss two major types of emergent cooperation among classifiers in a learning context.

Classifiers (i_1,i_2,o) and (j_1,j_2,p) are *chained* (or coupled) if they may fire in turn. For example, o may cause the environment to yield a message satisfying j_1, or the message m_o may satisfy j_2, or both. A successful chain of classifiers may recognize a situation and begin a series of actions leading systematically to large R. For these chains to emerge, the system must discover first, then maintain the needed classifiers. The bucket-brigade algorithm (BBA) is introduced to help these structures consolidate once they are discovered. The basic idea is that active classifiers must pay a fraction of their strength to those classifiers that in the previous time step either posted a message used by the classifier or just induced some action. The BBA creates in effect a flow of strength from the last classifier in the chain (whose strength is nurtured by incoming R) back to those classifiers implementing necessary though preliminary steps in the sequence of actions, with the result that all classifiers in the chain may reach similar levels of strength. Details on the BBA are provided later.

Default hierarchies constitute a second important type of emergent cooperation. In a simple example, (i_1,i_2,o) and (j_1,j_2,p) implement a default hierarchy if (i) conditions (j_1,j_2) are more specific (demanding) than (i_1,i_2), and (ii) the action induced by o is correct except when the second classifier is relevant also, in which case the correct action is given by p. Thus, when conditions (j_1,j_2) are satisfied, both classifiers are eligible for activation, so the system must favor the second (which will be reinforced) to protect the first from making a mistake (and losing strength). On the other hand, the first will be reinforced when (i_1,i_2) -- but not (j_1,j_2)-- are satisfied. This kind of structure seems to be achieved by letting the competition process depend more heavily on the *specificity* of classifiers, the ratio of specified to total number of coordinates: other things being equal, those classifiers that fit more closely the present context are preferred over more general alternatives. Default hierarchies can have arbitrary length and are relatively economical knowledge representation tools in some cases, see [Holland et al. 1986].

Both chains and default hierarchies have been observed in practice, although not overwhelmingly. Indeed, their formation and stability appear to be sensitive to certain system parameters, the effect of some of these parameters being only partially understood [Riolo 1987ab, 1989ab]. Even under the best conditions, the required time for the emerging structure to assemble solidly may be too large; therefore, certain ad-hoc mechanisms have been proposed to speed-up the process. A summary of results about emergence in CSs is provided in [Lane 1992], which also resumes the work of Riolo's [1991] lookahead CS not discussed here (see also [Holland 1990]). Other work on emergence in CS is provided by [Forrest and Miller 1990].

For ease of reference, it is useful to briefly outline the work of the GA. Typically, the GA is invoked after the system has been running for a while and strength has had some time to be reallocated. The entire population of classifiers (encoded as strings concatenating the classifiers' various schemata) is then renewed according to the following schedule. Two parent classifiers are selected with probability proportional to current strength and

either they are copied verbatim into the next population, or they undergo genetic manipulation (the latter occurring with certain probability). The standard manipulations are called crossover and mutation. Crossover randomly exchanges some coordinates from the two parent classifiers (choosing one or more cutting points in the strings), while mutation simply flips coordinates (with very low probability). The process is continued until a new population is completed. The success of the GA depends primarily on the extent to which the building-block hypothesis holds, that is, to the extent that high-performance strings (classifiers) are made up of "parts" that yield relatively good results no matter how the rest of the classifier is built; see [Mitchell et al. 1991] for additional discussion.

We close this section by pointing out a few key features of the general architecture. First, CSs exhibit parallelism at many levels, [Forrest 1991]; in fact, a parallel version deploying one classifier per processor has been implemented, see [Robertson and Riolo 1988]. Second, by ignoring the problem of effective activation at each step and looking at classifiers in a chain as pieces of code operating sequentially on some initial input to yield a desired output, chains can be conceived as subroutines performing standard operations (as eg. binary sum or set intersection). Indeed, CSs are shown to be computationally complete in the sense that a set of classifiers can be constructed to compute any function from the set of all possible message lists onto itself, [Forrest 1991]. Third, CSs are inherently pragmatic, easily implementable systems: they operate on a simple trial-and-error basis and do not require any kind of prior information about the task to be learned. Last, most qualitative aspects of the standard model have cognitive motivations, and, as shown below, a number of additional ideas inspired on cognitive processes can be incorporated easily. Hence, CSs may prove a useful tool in the computational analysis of cognitive phenomena.

3 Further ideas

Each of the following subsections deals with certain aspects of the work of CSs that have been subject to closer examination or criticism. The standard system is then analyzed and often modified or extended to cope with the perceived difficulty. The material below covers, in the author's opinion, some of the most suggestive pieces of research in CSs. Other ideas not presented here are discussed in [Holland 1987] and [Wilson and Goldberg 1989].

3.1 Difficulties with the GA

The first heuristic discovery mechanism considered in CSs is the genetic algorithm (GA). It was soon noticed, however, that the GA may not find the best conditions to operate within CSs. For one thing, CSs will need to search for *many* useful classifiers simultaneously, while the GA tends to converge to the best area of the search space. Also, heavily context-dependent strength might not provide a solid enough fitness function to guide the process. In fact, the computational theory of natural coadaptation described in [Holland 1962,1975] provides no direct guidance for the GA to discover *coadapted* classifiers in standard CSs, [Grefenstette 1987].

Among other remedies discussed in the literature, Smith [1983] presents a different type of CS, usually referred to as the P(ittsburgh) CS (the standard model is similarly associated with the Michigan school). In a nutshell, PCSs consider the entire *strategy* or population of classifiers as the main inferential unit. By stringing classifiers together, the GA is applied over strategies under the proviso that no classifier can be broken up by crossover. The level of fitness of each strategy is obtained through independent (possibly parallel) testing against the environment.

When cooperation between rules is fundamental for the success of the learning system, PCSs would seem to provide a more natural framework for the GA at the expense of increased computational cost. Hence, the primary problem in PCSs is to get the most out of its computational effort, that is, to determine what information (besides the basic performance indicator) should be transferred from each individual evaluation to the fitness function in order to improve the learning rate. Grefenstette's idea [1987] is to exploit the BBA's work during evaluations by physically bringing together high-strength classifiers within the strategy so they form a cluster less likely to be disrupted by crossover. The basic assumption is that coadapted classifiers ultimately tend to have similar levels of strength, which is true in the case of chains of classifiers under certain BBA parameters. At the end of each evaluation, classifiers are thus always rearranged according to their resulting strenghts before the GA is called, and new strategies containing clusters from successful strategies are thus more easily found. Note that nothing is said about the discovery of individual classifiers; in [Grefenstette et al. 1990], the preference is made explicit that this rather be taken care of by heuristic operators other than the GA.

The PCS approach has been implemented with success: Smith's LS-1 [1983] learned to play poker, beating a conventional knowledge-based system. Grefenstette's system RUDI [1987,1988] solved a difficult navigation problem where ultimate success was strongly dependent on the emergence of (long) chains of classifiers. In this problem, RUDI was shown to outperform standard CSs under various strength-revision mechanisms. On the other hand, there is some evidence that suitably modified CSs can achieve similar success rates, [Booker 1989], sometimes requiring far less resources, [Riolo 1991]. A clear-cut conclusion does not seem possible yet, as the modifications proposed by Booker and Riolo increase the system's computational complexity substantially and no scale-up analysis is available. As far as aiding the GA is concerned, Booker follows the idea of restricting mating among classifiers known to be related, thus attempting to curb the fraction of fruitless crossovers, see also [Goldberg 1989]. Booker stresses also the importance of triggering the GA when it is actually needed, that is, when the system somehow finds itself with "insufficient resources". These ideas are made more precise in the context of the next section, otherwise devoted to competition/reinforcement issues.

3.2 Competition and reinforcement

In the standard BBA, the auction and strength revision processes are conducted as follows: (i) classifiers scan the message list and environmental input for a perfect match. Only classifiers whose conditions are matched participate in the auction. (ii) Each matched classifier places a bid B, which is usually taken as $B=KS\delta(D)$, where K is a small constant ($\approx .1$) and δ is a nondecreasing function of specificity. Bids are further processed into effective bids $B^*=B^{\alpha}D^{\beta}$, where α and β are nonnegative constants. (iii) A fixed number of classifiers to be activated (say m) are selected without replacement with probability proportional to B*. (iv) The *bids* B of the m winners are subtracted from their strength; this strength is either paid to classifiers that posted the messages used by the winners or simply dumped out of the system. (v) The amount of reinforcement that enters the system next is distributed among winners.

The basic strength revision equation is $S_{i+1}=S_i-B_i+P_i+R_i$, where R_i and P_i denote respectively the assigned share of reinforcement and payments from other classifiers (that used the classifier's output message at the previous time step). Classifiers that bid but are not selected do not have their strength modified in principle, although tax may be (and

usually is) applied to all classifiers in the system to bring down the strength of classifiers that bid rarely.

Note that all strength transactions are local, so no extensive book-keeping is necessary. In the general case, however, the fact that selection depends in a complex way on the remaining classifiers in the population obscures analytical develoment, which seems possible only under some simplifications. For example, convergence results are provided by Arthur [1990] in the case of a fixed set of classifiers, no tax, random stationary reinforcement, $m=1$, $\delta(D)\equiv 1$, $P_i\equiv 0$, $\alpha=1$ and $\beta=0$; under these conditions, the system's asymptotic behavior (namely, classifiers are selected with probability proportional to their expected reward) is called probability matching (to be discussed in detail in 3.4). A more comprehensive mathematical framework is proposed in [Holland 1986b]. It is also worth noting that the BBA can be seen as a special case of Sutton's [1988] *temporal difference* learning algorithm.

The above framework leaves unresolved many practical issues. What is the precise role of D or α and β? How should reinforcement be distributed among winners? To what extent is it natural that correct classifiers not winning the competition be deprived of reinforcement? Are there faster ways to reallocate strength? How should the number of winners or the population size be chosen? Partial answers to these questions are put forward in the literature, although the viewpoints adopted often differ and colliding conclusions arise sometimes.

The rest of this section reviews an alternative auction/reinforcement scheme considered in [Wilson 1985,1987] and [Booker 1989]. Let us start by focusing on the role of m, the number of winners. That not all matched classifiers should be allowed to fire is clear (after all, we can not move left and right at the same time). When no contradictions arise, the main purpose of limiting the number of winners (or, alternatively, the size of the message list) is to force the system to focus on a few (critical) aspects at a time. Since selection is random, every classifier is nevertheless given a chance to be deployed. Note that winners are the only classifiers responsible for the system's actions *and* are the only ones eligible to have their strengths revised.

The alternative scheme proposed by Wilson and Booker differentiates these two functions. Their proposal is formulated within the particular context where (i) the system is expected to act at every step and (ii) no message list is used since all classifiers induce actions (however, it does not follow that $P_i\equiv 0$, as classifiers active at one time step may make payments to those classifiers active in the previous time step). This simpler architecture is often termed a *stimulus/response* (S/R) CS; the system to be introduced later in section 4 constitutes a simple example of this kind of system, but it will be seen to adopt the standard BBA. The key issue in (S/R) CSs is obviously how to determine the prescribed action. Instead of simply imposing $m=1$ in the standard auction, a subset of $m>1$ classifiers are still selected among a *cluster* of $M>>m$ *excited* classifiers (including classifiers matched exactly *and*, if these weren't enough, classifiers with best *partial* matches). Then, a single winner is picked at random among those m. Further, reinforcement is extended to *all* excited classifiers and depends on whether their suggested actions agree with the system's action and on the amount of reinforcement that such an action brought in.

Booker founds the role of the cluster and the derived reinforcement plan on ideas from stimulus sampling theory in psychology. The cluster is seen as a general pool of heuristics

trying to cope with a vast population of environmental stimuli. Whenever a reply is needed, the system in effect *samples* the cluster and the action is decided upon by sample majority. Therefore, the role of the set of m winners is to provide "an economical statistical summary of the cluster, not to hedonistically compete for rewards", [Booker 1989; p. 269].

Booker finds it useful to view the cluster as the basic inferential unit for various purposes. Indeed, he also proposes to administer the GA *locally* on the basis of a new quantity called significance. Each classifier is assigned its own significance reservoir, which is updated just like strength except that transactions involve completely different units. In particular, all classifiers in the same cluster are distributed a *fixed* amount of significance that is granted at each step. The share each classifier receives is proportional to its *effectiveness*, a notion similar to the effective bid that depends among other things on the strength and *relevance* (as measured by a partial matching score) of the remaining classifiers in the cluster. Therefore, only classifiers that are nowhere effective will tend to have low significance. In a simple implementation, Booker's focused sampling plan for the GA begins by triggering the GA precisely when a threshold level of low significance is achieved in some cluster. It then selects parents from the cluster, acting in the usual way except that nonsignificant classifiers get replaced by the offspring. The fact that deletion of classifiers is based on significance and not on strength seems to convey an improved idea of fitness, because now useful classifiers with medium or low strengths receiving only moderate reinforcement will tend to be significant and thus preserved in the system.

3.3 Long-term dynamics

It has often be the case in practice that families of identical or nearly identical classifiers evolve as the result of the GA's continued work on some successful classifiers and their offspring. It is both suggestive and customary to borrow terms from the ecological world to describe the various roles in this context. Each of such families may be called a *species*. Species are generated in the first place because there exists a source of payoff that supports them; this source of payoff may be called a *niche*. The environment provides directly a number of *primary* niches, all other niches are secondary. The amount of payoff that a given niche provides may be loosely called the *capacity* of the niche. The metaphor may go on: sharing payoff among winning classifiers corresponds to the idea of finite capacities; *parasitic* classifiers may emerge that don't belong to any species but enjoy payoff from one or more niches, etc. In terms of Booker's approach discussed in the previous section, a species can be equated to a (well-developed) cluster for a given niche.

Since niches will typically come with widely different capacities, niches with larger capacities will tend to be found and exploited first. Unrestricted proliferation of the corresponding species may lead then to a loss of diversity that may not leave room for secondary niches to be discovered or might congest the message list precluding effective communication (recall that the same message may be posted many times). For this reason, some authors have suggested the adequacy of limiting the maximum size of a species, that is, the number of replicates that may coexist, [Compiani et al. 1989].

Additional insights are gained by studying evolution of strength within a given species under controlled conditions, [Compiani et al. 1990]. These authors use the standard BBA with parameters $I(D)=1$, $\beta=0$ (thus excluding D from the process) and tax (at a rate not greater than 2%), and consider a species of μ identical copies of a successful classifier. Three patterns can be differentiated depending on m, μ and the effective bid parameter α. When $\mu=m=15$ and $\alpha=1$, all classifiers in the species are of course activated and reinforced

simultaneously and their strengths do reach rapidly the common fixed-point (asymptotic) level regardless of their starting values. This is called strength *equalization.*

When μ=10, m=2 and α=1, typically seven classifiers undergo complete decay to zero strength, while the other three classifiers "survive" and compete against each other. For each of them, the time elapsed between successive wins is a random variable, with the result of steady decay phases corresponding to longer periods of inactivity. On winning again, however, they recover enough strength so that they remain fully competitive (they "forget" the inactivity phase). Conversely, as long as they win repeatedly, their strength reaches the fixed-point level and it stays there until they lose the competition once. This is called strength equalization on the average or strength *oscillation.*

Finally, when μ=10, m=5 and α=2, and the bid constant K is set to .5, classifiers behave basically like in the previous case, except now the consequences of longer periods of inactivity are amplified due to the bias introduced by raising K and α. These periods of starvation bring strength to a point where it becomes extremely hard for classifiers to win the competition, with the result that five classifiers are eventually driven out to zero strength. When this occurs, the situation is basically that of the first case, so the strengths of the five survivors are equalized. These results are interesting since they indicate that, at least under some BBA parameters, the size of the message list may effectively control the spread of species. They also suggest that strength oscillation may be an important issue when the set of competing classifiers is allowed to vary.

3.4 Probability matching and reliability

Probability matching behavior (PMB) is a kind of suboptimal behavior (surprisingly exhibited by humans and other animals when exposed to an uncertain environment) to which much study has been devoted in the CS context, [Booker 1989], [Arthur 1990], [Goldberg 1990]. Arthur argues that PMB is not acceptable in an economic environment: for example, if, in the context set up in section 3.2, there are N=100 actions, the first one pays an average of 10 units and the rest pay an average of 10/11 units, then the action ten times better than the rest is activated only about 10% of the time. It would be better if the system could somehow distinguish a "clear" situation like this and choose more often or systematically the rule with highest expected payoff, a behavior that can be termed decision-theoretic (DTB). Arthur [1990] proposes a basic modification to the standard BBA (namely, that bids are not subtracted from the winners' strength) that does lead to DTB.

Goldberg [1990] argues that PMB might not always be an unreasonable choice. In essence, it is argued that what may make PMB a plausible guide for action from an evolutionary point of view is the possibility of *brisk* change in the environmental laws governing the short-term outcomes of our actions. PMB might then be adopted as a *prudent* alternative.

Goldberg [1990] studies mechanisms that allow CSs to switch from PMB to DTB depending on circumstances. Suppose l(D)=1 and tax is null. In [Goldberg 1989], the standard auction is modified in that (i) effective bids are computed as $\mathcal{B}$=B+τ, where τ is a N(0,v) variate, and (ii) winners are selected deterministically according to the highest effective bids. If the variances v are small, selection is conservative as higher bidders are more often selected; if they are large, bid differences are neutralized and selection becomes random. Somewhere in between, the system should implement PMB. By modifying v

dynamically, the system's desired flexibility may be obtained. The question is then centered on what constitutes a sensible basis for manipulating this variance.

Under this version of the BBA, the fixed-point bid is an estimator of average payoff. Goldberg [1990] therefore proposes to monitor the evolution of the difference $B-R_+$, where R_+ is the total amount of strength received from either the environment (R) or other classifiers (P). Specifically, each classifier maintains a *reliability* index $\rho_i(t)$, computed as a recent average of the observed differences $(B_i-R_{+i})^2$, that complements the job of strength by measuring the discrepancy between expected and actual payoff (a similar measure is introduced in [Booker 1989]). A *variance-sensitive* bidding scheme can be adopted by dynamically setting $v=\rho_i(t)$. When R_+ is fixed, B converges to it, and hence $\rho_i(t)$ will eventually approach 0. Thus, classifiers that consistently achieve payoff R_+ have effective bids converging to R_+. However, when there is inherent variability in R_+, $\rho_i(t)$ will approach a positive limit and the effective bid will fail to converge. A (S/R) CS endowed with this kind of bidding is shown to be able to shift rapidly from PMB to DTB (and viceversa) according to changes in the environment, [Goldberg 1990].

3.5 Long-term memory

Goldberg's CS ability to shift from one behavioral repertoire to another may also be seen as a weakness in that classifiers that implement the first repertoire may be completely swept out by those causing the second. In other words, the system may have to learn *again* the first set of classifiers should the conditions that promote the second persist long enough: CSs tend to be forgetful. It is convenient to distinguish between the kind of *short-term* memory played out by the message list (and the step-to-step linkage among classifiers) and the kind of *long-term* memory that would be required to store populations of classifiers not presently needed (thus enabling the system to benefit from the obtained expertise). For the latter type to be useful, the system should also have some means of identifying situations where the current population should give place to (fractions of) previously stored populations. These issues are investigated in [Zhou 1990], where it is proposed to enhance the standard architecture as follows.

Zhou's classifier system with *memory* (CSM) provides a model for cumulative learning that includes a long-term memory (LTM) buffer where *versions* of the rules that solve a given task are stored for subsequent use (rules stored in LTM are inactive unless recruited). Those versions summarize what has been learned at the end of a *learning phase* and are constructed by the *generalizer* algorithm. The work of the generalizer involves several processes: It first isolates a set of *solution* rules; this is achieved by rerunning the problem with the GA turned off so that the BBA can reallocate strength consistently and the strongest rules picked out. Next, it extracts common patterns by applying an intersection operator to the solution rules. The output of the intersection operator is further tested for correctness and redundant rules are eliminated. Finally, the resulting generalized rules are stored in a "chunk" in LTM. Chunks are organized in LTM into *domains* on the basis of both the nature of tasks and the generality of the condensed knowledge (see the original source for details).

Once useful knowledge has been conveniently generalized and stored, the question is how to make use of it. CSM uses transformational analogical reasoning ideas, [Carbonell 1983]: solution classifiers from previous similar tasks are recognized and transformed to apply to the present task. The recognition phase is primarily accomplished by the *matcher* algorithm, which is invoked at the beginning of a new learning phase (or whenever the

current rules in the CSM can not match incoming messages at an appropriate rate). The matcher implements a partial-match scoring function that measures similarity of tasks by contrasting conditions in stored classifiers with messages in the message list. By aggregating individual scores, chunks in LTM are ranked according to relevance in the current situation. The *initializer* algorithm works on the basis of this information and selects the most promising chunks. Classifiers in these chunks are finally modified somewhat to fit better the present context.

In simulations, CSM outperformed the standard CS in two series of robot navigation tasks requiring the use of coupled sequences of classifiers, [Zhou 1990]. It was first shown to be able to benefit from the solution of simpler problems by *transferring* this knowledge to solve more difficult problems. Success was also obtained in *reversal* learning, which is concerned with learning behaviors under sudden changes. Two tasks were given alternatively, switching the destination points while keeping the starting point fixed. By retaining the knowledge obtained in the first two trials, CSM was able to perform near optimally from the third trial on; the standard CS did not show much improvement even after 10 reversals, [Zhou 1990].

A practical limitation of CSM is introduced by the difficult problem of defining learning phases precisely, as standard CSs never needed them (they are not usually essential in human learning either). Stabilization of the (smoothed) learning curve (eg., a moving average of recent performance) may provide a general principle for applications. A different, hierarchical approach involving learning episodes is proposed in [Wilson 1989].

Endowing a CS with long-term memory is a profound and radical innovation. The traditional CS is rooted in the idea of *recency*: messages only last for one time-step unless reposted and strength is the only entity that reflects accumulated experience. From a cognitive or statistical point of view, this emphasis may be open to criticism. The system to be described in the next section incorporates mechanisms of an aggregative character that extend the basic architecture in a similar direction. Specifically, each classifier is granted the ability to remember a few cases where its output proved unappropriate. Thus, only *individual* learning episodes (controlling the amount of data to be stored by each classifier) are needed.

4 Data analysis

This section describes the basic ideas in PASS (Predictive Adaptive Sequential System), a simple classifier system (CS) for automatic exploratory data analysis, [Muruzábal 1992,1993]. PASS builds on the ideas expressed in [Packard 1989] and extends the S/R CS BOOLE and related systems, cf. [Wilson 1987], [Bonelli et al. 1990], in four major aspects. These can be summarized as follows.

First, the problem of predicting the unknown label y associated with an incoming input x is generalized from the dicotomous to the continuous case. Hence, classifiers in PASS share the basic structure

IF i THEN PREDICT p

where i is a schema condition and p is a (discrete) predictive probability distribution over the range of y. The subspace of probability distributions explored by the system is constrained only at the syntactic level (namely, all p have convex, relatively small support).

Second, classifiers that win the competition mix their individual predictions according

to strength to yield the system's overall prediction. Performance is measured on the basis of the predictive probability assigned to the true response. The BBA acts as usual on the strength of winners.

Third, a fixed number of exceptions are stored by each classifier in its *exception list* E. Exceptions are previously observed data pairs (x,y) such that x is in i but y does not belong to the support of p. When E is filled up, its contents are processed by various taylor-made procedures that may modify the classifier or create new ones. These procedures are based on simple heuristics and basic statistical measures of concentration.

Lastly, PASS applies a GA with restricted mating where crossover applies only to the schemata of classifiers with similar predictions. This provides an alternative to the more usual policy described earlier in section 3.2.

As discussed elsewhere, cf. [Muruzábal 1993], PASS has proven useful extracting regression knowledge in simulated streams. As a sequential belief-updating methodology, it can be compared to the bayesian approach, see [Muruzábal 1992], [Lane 1992]. Future work will investigate some of the research directions contained in this paper and possibly some other ideas in automatic nonparametric regression recently proposed in the machine learning literature.

References

[Arthur 1990]
W. B. Arthur. *A Learning Algorithm that Mimics Human Learning*. Working Paper 90-026, Santa Fe Institute, Santa Fe, NM.

[Bonelli et al. 1990]
P. Bonelli, A. Parodi, S. Sen and S. W. Wilson. *NEWBOOLE: A fast GBML system*. Proceedings of the Seventh International Conference on Machine Learning, Morgan Kauffman, San Mateo, CA.

[Booker 1989]
L. B. Booker. *Triggered Rule Discovery in Classifier Systems*. In J. D. Schaffer (Ed.) Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kauffman, San Mateo, CA.

[Carbonell 1983]
J. G. Carbonell. *Learning by Analogy: Formulating and Generalizing Plans from Past Experience*. In R. S. Michalski, J. G. Carbonell and T. M. Mitchell (Eds.) Machine Learning: An Artificial Intelligence Approach. Tioga Press, Palo Alto, CA.

[Compiani et al. 1989]
M. Compiani, D. Montanari, R. Serra and P. Simonini. *Asymptotic Dynamics of Classifier Systems*. In J. D. Schaffer (Ed.) Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kauffman, San Mateo, CA.

[Compiani et al. 1990]
M. Compiani, D. Montanari and R. Serra. *Learning and Bucket Brigade Dynamics in Classifier Systems*. Physica D, 42.

[Forrest 1990]
S. Forrest. *Emergent Computation: Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks*. Proceedings of the Ninth Annual CNLS Conference, Center for Nonlinear Studies, Los Alamos National Laboratory, NM.

[Forrest 1991]
S. Forrest. *Parallelism and Programming in Classifier Systems*. Research Notes in AI, Morgan Kauffman, San Mateo, CA.

[Forrest and Miller 1990]
S. Forrest and J. H. Miller. *Emergent Behavior in Classifier Systems*. Physica D, 42.

[Goldberg 1989]
D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.

[Goldberg 1990]
D. E. Goldberg. *Probability Matching, the Magnitude of Reinforcement, and Classifier System Bidding*. Machine Learning, 5.

[Grefenstette 1987]
J. J. Grefenstette. *Multilevel Credit Assignment in a Genetic Learning System*. In J. J. Grefenstette (Ed.) Proceedings of the Second International Conference on Genetic Algorithms. Lawrence Erlbaum, Hillsdale, NJ.

[Grefenstette 1988]
J. J. Grefenstette. *Credit Assignment in Genetic Learning Systems*. Proceedings of the Seventh National Conference on Artificial Intelligence. Morgan Kauffman, San Mateo, CA.

[Grefenstette et al. 1990]
J. J. Grefenstette, C. L. Ramsey and A. C. Schultz. *Learning Sequential Decision Rules Using Simulation Models and Competition*. Machine Learning, 5.

[Holland 1962]
J. H. Holland. *Outline of a Logical Theory of Adaptive Systems*. Journal of the ACM.

[Holland 1975]
J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI.

[Holland 1985]
J. H. Holland. *Properties of the Bucket-Brigade Algorithm*. In J. J. Grefenstette (Ed.) Proceedings of an International Conference on Genetic Algorithms and their Applications. Lawrence Erlbaum, Hillsdale, NJ.

[Holland 1986a]
J. H. Holland. *Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems*. In R. S. Michalski, J. G. Carbonell and T. M. Mitchell (Eds.) Machine Learning: An Artificial Intelligence Approach II. Morgan Kauffman, San Mateo, CA.

[Holland 1986b]
J. H. Holland. *A Mathematical Framework for Studying Learning in Classifier Systems*. Physica D, 22.

[Holland 1987]
J. H. Holland. *Genetic Algorithms and Classifier Systems: Foundations and Future Directions*. In J. J. Grefenstette (Ed.) Proceedings of the Second International Conference on Genetic Algorithms. Lawrence Erlbaum, Hillsdale, NJ.

[Holland 1989]
J. H. Holland. *Using Classifier Systems to Study Adaptive Nonlinear Networks*. In D. L. Stein (Ed.) Lectures in the Sciences of Complexity: Proceedings of the 1988 Complex Systems Summer School, Santa Fe Institute. Addison-Wesley, Reading, MA.

[Holland 1990]
J. H. Holland. *Concerning the emergence of tag-mediated lookahead*. Physica D, 42.

[Holland and Reitman 1978]
J. H. Holland and J. S. Reitman. *Cognitive Systems Based on Adaptive Algorithms*. In D. A. Waterman and F. Hayes-Roth (Eds.) Pattern-Directed Inference Systems. Academic Press, New York.

[Holland et al. 1986]
J. H. Holland, K. J. Holyoak, R. E. Nisbett and P. R. Thagard. *Induction: Processes of Inference, Learning and Discovery*. MIT Press, Cambridge, MA.

[Lane 1992]
D. A. Lane. *Artificial Worlds and Economics*. Technical Report # 582, University of Minnesota, Minneapolis, MN.

[Mitchell et al. 1991]
M. Mitchell, S. Forrest and J. H. Holland. *The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance*. Technical Report 91-10-046. The Santa Fe Institute, Santa Fe, NM.

[Muruzábal 1992]
J. Muruzábal. *A machine learning approach to a problem in exploratory data analysis*. Ph. D. thesis. University of Minnesota, Minneapolis, MN.

[Muruzábal 1993]
J. Muruzábal. *PASS: a simple classifier system for data analysis*. Submitted for publication.

[Packard 1989]
N. H. Packard. *A Genetic Learning Algorithm for the Analysis of Complex Data*. Technical Report, Center for Complex Systems Research and the Physics Department, University of Illinois at Urbana, IL.

[Riolo 1987a]
R. L. Riolo. *Bucket brigade performance I: Long sequences of classifiers.* In J. Grefenstette (Ed.) Proceedings of the Second International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum, Hillsdale, NJ.

[Riolo 1987b]
R. L. Riolo. *Bucket brigade performance II: Simple default hierarchies.* In J. Grefenstette (Ed.) Proceedings of the Second International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum, Hillsdale, NJ.

[Riolo 1989a]
R. L. Riolo. *The emergence of default hierarchies in learning classifier systems.* In J. D. Schaffer (Ed.) Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kauffman, San Mateo, CA.

[Riolo 1989b]
R. L. Riolo. *The emergence of coupled sequences of classifiers.* In J. D. Schaffer (Ed.) Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kauffman, San Mateo, CA.

[Riolo 1991]
R. L. Riolo. *Lookahead Planning and Latent Learning in a Classifier System.* In J. A. Meyer and S. W. Wilson (Eds.) From Animals to Animats: Proceedings of the First International Conference on the Simulation of Adaptive Behavior. MIT Press, Cambridge, MA.

[Robertson and Riolo 1988]
G. G. Robertson and R. L. Riolo. *A Tale of Two Classifier Systems.* Machine Learning, 3.

[Shu and Schaeffer 1991]
L. Shu and J. Schaeffer. *HCS: Adding Hierarchies to Classifiers Systems.* Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kauffman, San Mateo, CA.

[Smith 1983]
S. F. Smith. *Flexible learning of problem solving heuristics through adaptive search.* Proceedings of the Eight International Joint Conference on Artificial Intelligence. Karlsruhe, Germany.

[Sutton 1988]
R. S. Sutton. *Leaning to Predict by the Method of Temporal Differences.* Machine Learning, 3.

[Wilson 1985]
S. W. Wilson. *Knowledge growth in an artificial animal.* Proceedings of an International Conference on Genetic Algorithms and their Applications. Lawrence Erlbaum, Cambridge, MA.

[Wilson 1987]
S. W. Wilson. *Classifier Systems and the Animat Problem.* Machine Learning, 2.

[Wilson 1989]
S. W. Wilson. *Hierarchical Credit Allocation in a Classifier System.* In M. S. Elzas, T. I. Oren and B. P. Zeigler (Eds.) Knowledge Systems' Paradigms. North-Holland, New York.

[Wilson and Goldberg 1989]
S. W. Wilson and D. E. Goldberg. *A Critical Review of Classifier Systems.* In J. D. Schaffer (Ed.) Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kauffman, San Mateo, CA.

[Zhou 1990]
H. H. Zhou. *CSM. A Computational Model of Cumulative Learning.* Machine Learning, 5.