Working Paper 93-20
Statistics and Econometrics Series 16
September 1993

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

PASS: A SIMPLE CLASSIFIER SYSTEM FOR DATA ANALYSIS

Jorge Muruzábal*

Abstract

Let x be a vector of predictors and y a scalar response associated with it. Consider the regression problem of inferring the relantionship between predictors and response on the basis of a sample of observed pairs (x,y). This is a familiar problem for which a variety of methods are available. This paper describes a new method based on the classifier system approach to problem solving. Classifier systems provide a rich framework for learning and induction, and they have been successfully applied in the artificial intelligence literature for some time. The present method enriches the simplest classifier system architecture with some new heuristic and explores its potential in a purely inferential context. A prototype called PASS (Predictive Adaptative Sequential System) has been built to test these ideas empirically. Preliminary Monte Carlo experiments indicate that PASS is able to discover the structure imposed on the data in a wide array of cases.

Key Words
Regression analysis; Classifier systems; Machine learning.

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

PASS: A simple classifier system for data analysis

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

Abstract

Let x be a vector of predictors and y a scalar response associated with it. Consider the regression problem of inferring the relationship between predictors and response on the basis of a sample of observed pairs (x,y). This is a familiar problem for which a variety of methods are available. This paper describes a new method based on the classifier system approach to problem solving. Classifier systems provide a rich framework for learning and induction, and they have been successfully applied in the artificial intelligence literature for some time. The present method enriches the simplest classifier system architecture with some new heuristics and explores its potential in a purely inferential context. A prototype called PASS (Predictive Adaptive Sequential System) has been built to test these ideas empirically. Preliminary Monte Carlo experiments indicate that PASS is able to discover the structure imposed on the data in a wide array of cases.

Keywords: Regression analysis; Classifier systems; Machine learning.

1 Introduction

Let x be a vector of predictors and y a scalar response associated with it. Consider the regression problem of inferring the relationship between predictors and response on the basis of a sample of observed pairs (x,y). This is a familiar problem for which a variety of methods have been proposed in the statistical, machine learning and neural network literatures. Such methods can be categorized according to a number of basic dimensions, as eg. type of processing (batch vs. sequential), purpose (estimative vs. predictive) or inferential engine (data-driven vs. model-driven). Given the increasing volume of raw information and the advent of powerful computing systems, fully automatic approaches have received

[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, New Mexico, and by grant DGICYT PB92-0246 (Spain).

much attention in recent years; hence computational complexity and degree of inherent parallelism are important considerations as well. Finally, since one of the goals of the analysis is to better understand the underlying data-generating process, developers must also be concerned with output interpretability.

This paper presents a new method based on the classifier system approach to problem solving, [Holland 1986,1989]. The new method is described in detail and put in perspective with respect to previous work in classifier system research. A prototype called PASS (Predictive Adaptive Sequential System) has been built to test the ideas empirically. Preliminary experimental results suggesting the usefulness of the method are outlined, and some of the weaknesses and strengths of the prototype pointed out.

PASS has been designed to solve the particular problem in which x is a *boolean* vector of predictors (sometimes called the *stimulus*) and y varies over the unit interval (0,1). Only this *canonical* version of the problem is considered throughout. Since any real number can be approximated by a binary expansion, continuous predictors can be handled in principle to any degree of accuracy, although the encoding procedure is open to some subtleties (see below).

2 Classifier systems, genetic learning, data analysis and PASS

This research borrows from and extends a simple classifier system (CS) architecture, [Holland 1986,1989]. Since many excellent works on CS theory are available, only a brief general introduction is provided here. For a thorough exposition including philosophical foundations, the reader is referred to the monograph [Holland et al. 1986].

In general, a CS models the process of adaptation of an organism or agent immersed in some environment. Organism and environment are continuously interacting: on one hand, the organism "perceives" a number of signals arising from the environment and reacts to them; on the other, the environment behaves as a dynamical system that may be partly affected by the organism's actions. Adaptation is the process whereby the organism learns a set of useful behavioral rules on the basis of all previous experience. At any given time, the set of rules entertained by the agent constitutes his current model of the situation: it provides him with a script specifying what to expect or what to do when certain environmental patterns are perceived. Rules are useful in that they lead to the agent's *goal*, namely, the (frequent) acquisition of *reward*, a special signal released by

the environment when some critical states are reached. In essence, the agent learns by a simple trial and error heuristic search: those rules in the model that prove useful are reinforced and maintained, so that they will be used again in future similar situations. Rules that do not yield the desired results are weakened and eventually discarded, leaving room for other plausible alternatives generated by the system's discovery mechanisms.

Genetic learning is commonly understood as the branch of machine learning which essentially encompasses genetic algorithms (GAs) and CSs. The theory of the GA, based on simple selection and recombination ideas, was developed earlier in [Holland 1975]. The GA has since become widely recognized as a powerful optimization tool, and a relatively large body of research is available on both theory and applications, see eg. [Davis and Steenstrup 1987] and [Goldberg 1989]. CSs make use of the GA albeit with a different purpose, namely, to provide a heuristic basis for the generation of new rules. However, its usefulness here has been subject to cogent criticism, cf. [Grefenstette 1987]. The main difficulty stems from the fact that the GA pursues a *single* optimal structure, while a typical CS looks for several target rules simultaneously. Thus, there is in principle no mechanism to prevent recombination of unrelated material. While some authors have suggested useful variations aimed to overcome this problem, cf. [Goldberg 1989], [Booker 1989], the overall effect of the GA within a CS seems far from understood, [Robertson and Riolo 1988], [Wilson and Goldberg 1989]. We will return to this issue later in the context of PASS. The point stressed here is that additional exploration heuristics should be much welcome in CS research; one such a source of "inspiration" is proposed below.

In order to apply the CS architecture in a data-analytic context, we begin by associating the learning system with the statistical analyst and the environment with a stream of data. Then, we must inquire about both goals (what is it exactly that the system should learn to do?) and interaction channels (in what ways can the system affect the environment's behavior?). These notions are not unrelated: the richness of the interaction channel evidently constrains the range of goals available to the system. Furthermore, some statistical goals may be in conflict with the basic principles underlying the work of CSs (eg. CSs do not remember past observations explicitly, neither do they assume any kind of structure in the data source).

An essential goal in all CSs is that of reducing uncertainty as much as possible by providing predictions of environmental response. PASS focuses on this idea by identifying the amount of reinforcement with predictive accuracy: the system is rewarded if and only if it is able to learn patterns in

the data stream that permit it to anticipate unseen responses. The more precise the (successful) prediction, the larger the reward.

Among the various ways to design interaction channels between the learning system and the data stream, the simplest approach is to assume no interaction at all: the data stream is unaffected by any actions taken by the learner. Among such streams, the natural starting point consists in assuming a random sampling scheme. This scheme, quite common in machine learning and statistical research, currently constitutes a central assumption in PASS. Under the random sampling assumption, the environment consists of an (infinite) stream of independent, identically distributed vectors (x,y) drawn from some joint distribution. Time-dependent data may also be treated within this framework (see eg. [Packard 1989]), but this possibility is not explored here.

We are now ready to examine the system's basic cycle. PASS processes data pairs (x,y) one at a time. At each step, it first looks at the stimulus x, and its short-term and sole goal is to predict the associated response y. The system's model consists of a set of classifiers or elementary predictions for certain classes of stimuli. The current stimulus selects the subset of *relevant* classifiers through the usual (exact) matching process. These classifiers compete with each other, with the result that only a fraction of them are responsible for the system's prediction. This prediction is then contrasted with the response y. Reinforcement may be applied to individual classifiers in various well-known ways, although some schemes seem to exhibit more difficulties than others. So far, nothing is very different from other so-called stimulus/response CSs. However, two novel features characterize the present approach. First, the system is evaluated according to the observed predictive efficacy: the higher the predictive probability assigned to the response, the better. Second, classifiers are granted the ability to *remember* a few observations temporarily, and this local memory forms the basis for the generation of new classifiers or the modification of existing ones. Each of these subprocesses is described in detail next. We begin by clarifying the nature of the present type of classifier.

3 Knowledge representation in PASS

PASS represents knowledge as an unstructured, evolving collection of elementary predictions called classifiers. The number of classifiers in the model is a system parameter; it will be suggested to some extent by the estimated degree of complexity in the data. Selection of the appropriate size in each case is crucial, for it affects the computational complexity (both

time and spacewise) of the algorithm, yet remains an open research issue. It has been suggested, [Robertson and Riolo 1988], that the larger the population size, the better (a conclusion that has been reached in PASS aswell). However, Riolo [1989] provides some conflicting insights on the effects of population size on the system's qualitative behavior.

All classifiers in PASS have the following structure:

IF s *THEN PREDICT* (d,f) {with *strength* S and *recalling* E},

where s is a hyperplane or *schema* in the boolean hypercube $\Omega=\{0,1\}^n$, (d,f) encodes a predictive distribution, S is a scalar reflecting the previous usefulness of the classifier and E is a list storing a few observations where the classifier "failed" (in a sense to be made precise shortly). Hyperplanes are represented as vectors whose components are 0,1 or ?, the "don't care" character (the more common symbol # has a special meaning in LISP). As usual, schemata from different classifiers may and are expected to overlap. An important feature of any given classifier is the *specificity* D=D(s), the proportion of non-? in the schema s.

Predictive distributions may be designed in various ways amenable to symbolic heuristic manipulation. At present, they are simply configured in terms of d, a boolean vector of arbitrary length (the support of the distribution), and f, a vector of probabilities over d. The coordinates of d correspond to an arbitrary number of equally-spaced subintervals of the unit interval; every support is thus characterized as the union of those subintervals tagged by 1's in d. The length of d in bits is called the *resolution* of the system. This parameter remains fixed throughout, although future versions may accommodate dynamic resolution (eg. *zooming*). Another quantity of interest is the number of 1's in d divided by the resolution; this is called the *scope* of d, and denoted by $\upsilon=\upsilon(d)$.

The reason why predictions are split into two components is operational: supports d, as well as schemata s, are amenable to direct manipulation by the system. This is not the case with f, which is always linked to d and otherwise beyond the system's control. By default, classifiers start off with a uniform prediction over d. Later, as relevant data are recorded, the coordinates of f are updated according to the observed frequencies. For example, the classifier

```
(0??1??01) -> ((0000001111),(.1,.1,.4,.4))
```

(where redundant 0's in f have been deleted) represents the working

hypothesis that responses associated with stimuli satisfying the given schema lie on (.6,1) and tend to concentrate on (.8,1). The exception list contains observations (x,y) previously processed by the classifier such that x is in s but y is not in d. For example, the exception list to the above classifier might include data {(00010101), 0.23952}, {(01111001), 0.58637}, etc.; the information in the exception list is *not* used to build predictions.

Besides the choice of resolution, two other constrains operate on the population of classifiers. More precisely, these constrains limit the space of predictive distributions entertained by the system. First, only *convex* supports are allowed (ie. those vectors d whose 1's are not isolated). This policy enhances the overall interpretability of the output and considerably simplifies the design of support-manipulating routines, but it complicates the representation of bimodal patterns in the data (see below). Second, both upper and lower bounds act on scope. The lower bound means no serious limitation, as sharp predictions may always emerge within broader supports. On the other hand, larger scopes may impair decisively the flow of exceptions. The onset of an upper bound on scope introduces a built-in bias against patterns in the data that are too diffuse to "fit" in one classifier: it assumes the existence of less diffuse patterns which become the actual targets of the search.

The resulting knowledge representation is reminiscent of (and was partly motivated by) that in [Packard 1989], where, loosely speaking, constrains on support are replaced by an entropy-based explicit evaluation of uncertainty. However, Packard's treatment relies on batch-processing of a fixed data set and all learning is accomplished via the genetic algorithm. In contrast, the present, more flexible CS approach proceeds sequentially, allows other forms of learning and does not preclude the emergence of structure within the population of classifiers.

4 Prediction, evaluation and reinforcement

Once the system is provided with an initial set of classifiers, exposure to the data stream may begin. This paragraph explains how the system's predictive distribution is built, how performance is measured and how reinforcement can be applied. A few departures from the standard architecture are singled out. Some specific choices for system parameters are discussed later.

As usual, all classifiers in the current population are checked for a

match with the incoming x. If there is no match at all, the observation (x,y) is stored in the *no-match list* and no further action is taken. The no-match list is searched by procedure `scan` when it reaches some user-specified length, with the result of possible addition of classifiers from time to time. Procedure `scan` is the primary heuristic tool in PASS and is described in the next section.

As soon as there are one or more matches, the system enters the bidding/auction process. Each matched classifier bids an amount

$$b \propto l(D)\, S,$$

where D is the classifier's specificity defined earlier, l(D) is some nondecreasing function of specificity (typically taken to be either the identity or a constant) and S is current strength. The size of the bid determines how likely is a given classifier to be used by the system. By selecting a subset of *winners*, the system seeks to concentrate on the most useful components of the current model. On the other hand, the auction is stochastic, so all matched classifiers have a positive probability of winning. The number of winners (say m) is usually fixed throughout.

If there are less than m matched classifiers, all of them win and no competition is needed. Otherwise, *effective bids* are computed as $B = b^{\alpha}D^{\beta}$, and winners are chosen with probabilities proportional to B, that is,

$$\phi_i = P(i \text{ wins}) = \frac{B_i}{\sum_j B_j} .$$

Parameters α and β are nonnegative and have been shown to play an important role in both the stability of some emergent structures and the asymptotic behavior of families of similar classifiers, cf. [Riolo 1987] and [Compiani et al. 1990] respectively.

Winners *pay* for the right to guide the system by having its bid deducted from its strength; the remaining matched classifiers do not have their bids subtracted, but are not eligible for reward either. In addition to bid substraction, tax is collected at every step from *all* classifiers in the model. Tax is just a fixed fraction of strength needed primarily to reduce the strength of classifiers that are matched rarely if ever. Unless taxes are effective, the system has no way of getting rid of these useless classifiers.

The system's predictive distribution, say f^*, is computed as the mixture of the individual winning predictions weighted by strength:

$$f^* \propto \sum_i S_i \, f_i.$$

We now motivate the evaluation measure used in PASS. Given a collection of responses $\{y_i\}$, the standard measure of predictive efficacy is provided by the joint probability assigned to the collection; under the assumption of independence, this becomes the product of individual probabilities $f^*(y_i)$, ie. the probability assigned by f^* to the subinterval containing y_i. If the contributions from different responses are to be added together (eg. to obtain a moving average), performance should be based on the logarithms of these probabilities. However, since the support of f^* need not cover the whole unit interval, the central measure of performance in PASS is defined as

$$Y(f^*,y) = \tau \, [\, \kappa + \max\{\log f^*(y), -\kappa\}],$$

if $f^*(y) \neq 0$ (0 otherwise), where $\kappa > 0$ is a truncation parameter and $\tau > 0$. Note that Y is 0 until $f^*(y)$ reaches above $e^{-\kappa}$, then $Y > 0$ and increases with $f^*(y)$. Parameter τ is clearly irrelevant for evaluative purposes and could be set to 1 here; its role is discussed below. Selection of κ may be guided by examining specific distributions. For example, assuming a resolution of 16 bits, the uniform distribution over the whole interval assigns probabilities $f^*(y)=1/16=.0625$, whose logarithm equals -2.77, while the uniform distribution over the correct half assigns probabilities $f^*(y)=1/8=.125$, whose logarithm equals -2.08. In the experiments reported below, κ and τ were set to 3 and 100 respectively.

Let us finally discuss reinforcement and related issues. Depending on whether one focuses on partial or overall success as the basic criterion for reinforcement, two major regimes, called *individual* and *ecological* respectively, can be considered. We first discuss the former, which has proven most useful in practice, and briefly comment on the latter (a priori more congruent with the CS approach).

Under individual reinforcement, each winner receives a reward that depends solely on its own prediction: a simple boolean measure of success is introduced (namely, d captures y), and predictions are categorized as success or failure. If successful, winning classifiers are granted a reward

$R(\upsilon) \propto \upsilon^{-\gamma}$, with $\gamma \geq 0$. No reward is given if the classifier is not successful, and a fraction of strength, say ρ, may be deducted as a penalty. Most experiments have been based on $\gamma=1$ and $\rho=0$, although $\gamma=0$ leads to similar results and $0<\rho<.2$ has proven useful.

The previous measure of individual success has a second function in PASS, which applies to all matched classifiers regardless of reinforcement regime. Specifically, if d captures y, then y is used to update f, else (x,y) is stored in the classifier's exception list. Exception lists have limited capacity and are eventually passed on to the `explain` routine to search for regularities and modify the classifier base accordingly.

Updating individual predictions is done as follows. Strictly speaking, classifiers do not store f but F, which is a counter vector ordinarily initialized at d (some of PASS's heuristic operators may reinitialize F differently; in particular, the coordinates of F are usually but not always integer-valued). If d captures y, the corresponding coordinate in F is increased by 1; whenever f is needed, it is computed from F by normalization. No reinitialization prevents f from approaching the (truncated) empirical distribution function based on all subsequent data filtered by s (thus, the initial F is always noninformative in that it is soon overwhelmed by data). Of course, more formal updating procedures are conceivable.

Under ecological reinforcement, all winners receive the same reward depending on the system's success (regardless of their individual behavior). In PASS, this reward is precisely Y, the measure of performance defined above. Parameter τ thus controls the *intensity* of reinforcement; it should be large enough to allow good classifiers to depart from the initial strength level rapidly.

Although ecological reinforcement appears a priori more likely to support cooperative phenomena, experiments show that it brings two adverse effects. First, good classifiers are no longer guaranteed a fair reward because their predictions are often "contaminated". Second, poor alternatives (parasites) are sometimes systematically selected along with correct classifiers and thus prosper unduly. As a result, the distribution of strength over the population is more uniform and learning is impaired. To be useful, ecological reinforcement probably needs to be adjusted in order to reflect also individual performance.

5 Heuristic operators

The strength revision mechanism just described will eventually allocate strength so that the system exhibits the best possible performance given a *fixed* set of classifiers. To improve further, new classifiers are injected by the system's rule-discovery heuristic operators as described in this section. A few comments are in order before we review each operator in turn.

Operators perform a variety of transformations on classifiers; classifiers may have their schemata s or predictions d modified, and new classifiers may be generated from one or more existing classifiers. When a classifier is modified, the old version disappears, that is, transformations in PASS are irreversible (some restrictions might be considered for the sake of efficiency). When new classifiers are generated, some classifiers must be deleted since the population size is limited. In PASS, exiting classifiers are selected in a deterministic way among those having lowest strengths. However, since no classifier should be lost without a fair chance to be tested, a minimum *age* is set for deletion (eg. 75 observations); as a result, the system may occasionally exceed the maximum number of classifiers (which thus acts as an attractor). Finally, critical design decisions concern the strength or prediction to be attached to new or modified classifiers; these are presented later.

Procedure `scan` takes a list of I observations (x,y) and returns a list of classifiers based on concentration of response. `scan` can be used in three different contexts, depending on the nature of the input list: it can work on either a training sample in batch mode (thus providing a starting population; this option has not been used much), the no-match list or a classifier's exception list. The essential mechanism is the same regardless of context, and is described as follows.

`scan` first sorts out the data according to their y values. It then scans through the list of sorted y-values by sliding a window of user-specified length w=w(I) (taken as 1.5/I). As soon as V (eg. 5) or more observations are found within the window, `scan` signals a concentration of response and tries to characterize it. It will first check whether the enclosed x-values show common coordinates. If so, it manufactures a new classifier whose schema contains such coordinates and whose prediction (approximately) reflects the location of the concentration; classifiers constructed in this way are termed S-classifiers. If there are no common coordinates, `scan` ignores the concentration and continues scanning.

It will often be the case that some (or all) of the I observations in the

input list are not used in the generation of S-classifiers. As an attempt to avoid loss of useful information, `scan` also generates a tunable number of *G-classifiers*. Each G-classifier is based on a single unused observation: PASS simply generalizes at random the stimulus x and builds a prediction d by *expanding* (again, at random) the coordinate that contains the response. The idea is similar to the so-called Cover Detector operator, [Robertson and Riolo 1988].

Procedure `explain` is designed to exploit certain regularities in a classifier's exception list. It can modify existing classifiers or add new classifiers. It is triggered by currently *useful* (as measured by strength) and *improvable* (as measured by the estimated probability of individual success) classifiers when their exception lists reach certain critical length (say e_1). Thus, classifiers that are currently strong are favored, while classifiers that cover a substantial amount of mass may not be reasonably improved, so `explain` ignores them.

When `explain` acts on this set of e_1 exceptions, it first finds out whether or not the x-vectors exhibit common coordinates beyond those in the classifier's schema and whether or not the corresponding y-values tend to *cluster* (at present, the y-values are said to cluster whenever their range is less than some constant; more robust criteria may also be considered). On the basis of this information, `explain` branches into three cases, labeled `F` (no common coordinates), `N` (common coordinates but no cluster) and `E` (common coordinates and cluster). Different actions are taken in each case as follows.

In case `F`, `explain` simply stretches the classifier's prediction, an action that it will execute unless d is not too large (in terms of the upper bound on scope) after a new bit is added to it. As an example, consider the summary:

```
1 ????????0?0? 0000000000011100
2 ????????0?0? 0101000000000011
3 ????????0?0? 0000000000011110
```

(Line 1 contains the classifier s -> d as it stands before the transformation applies, line 2 shows the location of the exceptions (it also shows that no extra common coordinates were found among them), and line 3 portrays the transformed classifier).

In case `N`, `explain` thinks that the original classifier is too general and tends to specialize it. It first checks whether the classifier's schema is general enough ($D < .25$). When that is the case, s is augmented by

negating a randomly chosen new common coordinate; otherwise, no action is taken. Other variations are equally considerable: see cases **A1** and **A2** below. To illustrate (read lines as before):

```
1 ?0????????1 0000000000111000
2 ?0?0???1??1 1110000000000001
3 ?0??????0??1 0000000000111000
```

In case **E**, `explain` creates a new classifier (called an **E**-classifier) depicting the new regularity: s incorporates the set of common coordinates (or a random subset so that the new specificity remains below some constant), and d reflects approximately the location of the cluster. The following example is interpreted similarly, except that line 3 contains now the specialized version to be *added* to the model:

```
1 ??10????1?? 0000000001111000
2 1010????1?? 0000000000000111
3 1010????1?? 0000000000001111
```

Whenever a classifier is modified or a new classifier created, the old classifier's exception list is emptied. When no classifier is modified or generated at this stage, `explain` does not erase the exception list and simply quits. The exception list will then grow until a new threshold length is reached, say e_2, where `explain` is invoked again (and the same selection filters applied as above). This time, however, `explain` passes the exception list directly on to procedure `scan`, whose ordinary output is further processed as follows. `explain` first removes those **S**-classifiers whose schemata do not enhance the original schema. Second, for each remaining **S**-classifier (if any), `explain` checks whether its prediction is not *far away* from the original classifier's prediction (at present, this is simply determined by the number of bits between predictions: if there are more than a threshold number of bits, then predictions are considered far away. Of course, this threshold depends on the system's resolution and maximum scope).

When predictions are far away, the **S**-classifier is left untouched. Otherwise, `explain` replaces the **S**-classifier with a new classifier (called an **A1**-classifier) that combines the **S**-classifier's schema with the original prediction. It also introduces a second type of classifier, called an **A2**-classifier, that combines the old schema with the new prediction. Any **G**-classifiers produced by `scan` are returned in either case. Before exiting, the exception list is emptied (even if `explain` failed to produce new classifiers), so there are never more than e_2 observations in any exception

list. The following are some examples; the various lines show the original classifier along with the labelled addition(s) in each case:

```
  ?10???1??1? 0000000000111110
1 ?10???1??11 0000000000111110 A1
2 ?10???1??1? 0000000000000111 A2
3 ?10??11??1? 0000000000111110 A1

  01?????0??? 0000000000001111
1 011??700??? 1110000000000000 S

  ??10????0?? 0011110000000000
1 ????0????10 1110000000000000 G
2 0110???00?? 0000000000111110 S
```

PASS uses also two additional operators, called GA (a particular genetic algorithm) and `intersect`. Unlike `explain`, GA and `intersect` are invoked periodically (eg. every 250 observations) after the system has been running for a while, so each classifier has had some time to be tested and strength has been allocated accordingly. While the GA and `intersect` routines work independently of each other, they are invoked simultaneously because their mode of operation is similar. Specifically, both restrict mating among classifiers that are known to be related (see section 2). In PASS, classifiers are related because either their schemata are similar (so they tend to respond to the same stimuli) or their predictions point to the same region in the unit interval. Only the latter type of liaison is currently implemented, although experience shows that it is not very helpful (see below). Previous success with the former (see [Booker 1989]) suggests that it is probably a better alternative.

Here are the details of the mating policy. The first parent classifier is selected with probability proportional to current strength (but, to provide some focus, only among those classifiers with above median strength). Now the unit interval is split into a smaller number of regions (say 4 or 5). The second parent classifier is then selected at random among those classifiers whose predictions lie on one of the regions addressed by the first parent classifier. For example, if the resolution is 16 and the unit interval is split into 4 regions, the prediction `(0011100000000000)` addresses the first two of them.

GA renews about 8% of the population per activation. Following conventional practice, standard single-point crossover and mutation are included in GA; they apply to the schema part of classifiers only. Mutation

rate is set to .5%. One of the offspring schemata is randomly selected with either of its parents' d as initial prediction. To illustrate, consider the following example (the first two lines correspond to the parents and the third to the offspring respectively):

```
1 ??0???1??1? 0000000000011111
2 0?1??0????? 0000000000111110
  0?0???1??1? 0000000000111110
```

`intersect` also renews about 8% of the population. This procedure does exactly what its name suggests: it intersects both schemata and predictions and returns what it found in common in the parent classifiers. Intersected schemata intend to distille the most general regions in stimulus space, while intersected predictions try to focus on the center of the target. No empty schemata (or predictions) are permitted; `intersect` predictions are expanded if the resulting scope is too low. Here is an example (same reading as before):

```
1 ?011??????0 0000000001111100
2 111???0???0 0000000000011111
  ??1???????0 0000000000011100
```

Classifiers arising from `GA` and `intersect` are called `X` and `I`-classifiers respectively.

Now that all classifier types have been defined, let us examine the strength and prediction assigned to them (see Table 1).

Table 1. PASS initialization procedure

Type	Strength	Prediction
R,U,S,G,E	S_0	f_0
F	S	$(f+f_0')/2$
N,A1	S	f
A2	S	f_0
I	average	f_0
X	average	f

Here, `R` stands for randomly-generated and `U` for user-input. S_0 denotes both the user-input initialization strength (for types `R` or `U` in the initial population) and the median strength of the current list of classifiers (for

types S, G, E to be introduced later).

Classifiers R, U, S, G and E have their predictions initialized at $f_0 = f_0(d)$, the uniform distribution over d. Classifiers F,N,A1 and A2 all arise from a single classifier and maintain its strength S (an alternative is to set their strength to the dynamic S_0). Classifiers N and A1 maintain also their prediction f. Classifier F mixes the old prediction with f_0' -- the uniform distribution over the augmented support. The strength of X and I classifiers is the average of their parent classifiers.

6 Performance

PASS is implemented in LISP-STAT, [Tierney 1990], occupies about 104K and currently runs on a DEC-station 5000. The code is largely experimental in that no optimization effort has been made, so attention is focused on performance regardless of execution times. All experiments discussed below are based on simulated data. For the sake of brevity, only major results are outlined here; additional results are provided in [Muruzábal 1992].

Data (x,y) have been generated in two ways. The first and largest series of experiments refers to a mixture distribution on $\{0,1\}^n \times (0,1)$ (sometimes called the environment) involving $c \geq 1$ *concepts*, each specifying a particular regularity to be observed from time to time. Note that the present notion of concept is slightly more general than the one usually considered in concept learning (where it is understood as a subset of stimulus space). In particular, concepts are defined by triples (q,h,u), where $0<q\leq 1$ is the mixing proportion, h is a schema, and u is a distribution on (0,1). Without loss of generality, schemata h_i are assumed disjoint (this seems a minimum coherency requirement and permits straightforward calculation of the optimal level of performance attainable by the system). The data source simply extends so far the standard paradigm in boolean learning, cf. [Wilson 1987]. If possible, however, the set of concepts is automatically augmented with *noise*; noise is devised as a special triple (q_0,h_0,u_0), where u_0 denotes the uniform distribution over the unit interval and q_0 and h_0 represent respectively the complements of the sum of q_i and the union of h_i. The resulting distribution can be seen as a particular case of the "signal vs. noise" paradigm (see eg. [Titterington et al. 1985; section 6.3.2]). Sampling proceeds then as follows: a triple is selected at random according to the relative frequencies q_i, (i=0,1, ..., c), x is obtained by either filling up the undefined coordinates in h_i with

randomly chosen 0's and 1's or simply choosing a string from h_0, and y is taken as an independent realization of u_j. A wide array of situations can be considered by varying the amount of noise, the number of concepts, the specificity of schemata h and the sharpness of distributions u. The model allows also for other types of regions h_j, but these have not been investigated yet.

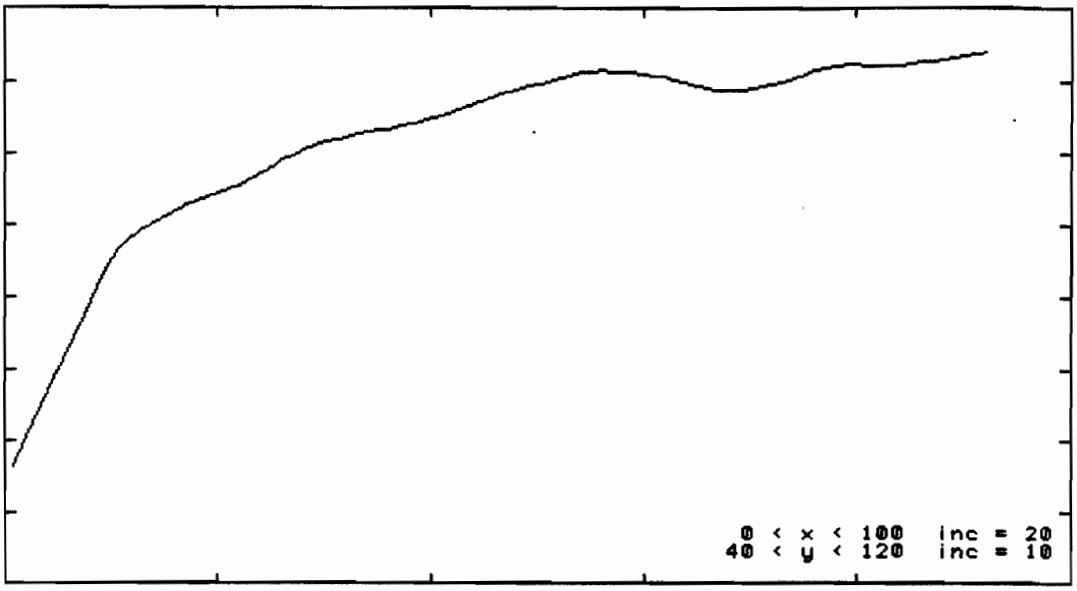
In evaluating the system, I first consider an environment of moderate difficulty and analyze the system's behavior in detail; a *baseline* performance level is thus obtained. I then modify the environment in various ways and compare performance with respect to the baseline in each case. The environment used to determine baseline performance is called M11 and is shown in Table 2 (for convenience, all distributions u are taken to belong to the Beta family and are specified by their respective parameters).

Table 2. Conceptual parameters (h,q,u) in the environment M11. M11 is a stochastic generalization of the well-known multiplexer problem, [Wilson 1987]).
Noise rate is 4%.

```
1 010??1?????  0.12 (5 1)
2 101?????0??  0.12 (1 5)
3 101?????1??  0.12 (5 1)
4 111????????0  0.12 (1 5)
5 0000???????  0.12 (1 5)
6 110??????1?  0.12 (5 1)
7 011???0????  0.12 (1 5)
8 011???1????  0.12 (5 1)
```

M11 has been studied under the following system parameters: resolution is set to 16 bits, and lower and upper limits on scope are .15 and .32 respectively. Auction parameters are m=5, l(D)=D, α=1 and β=3; a tax rate of .5% is in effect. Reinforcement is individual. Population size is set to 60 classifiers. Thresholds for exception lists are e_1=5 and e_2=15. Performance seems fairly robust under one-at-a-time variations of these parameters. Seven runs of 7,000 trials were conducted, and, for each run, the average performance Y was recorded every 75 trials. The median over runs was then smoothed according to the LOWESS algorithm, [Tierney 1990]. Figure 1 shows the resulting learning curve.

Figure 1
Median performance over 7 runs of 7,000 trials.



This figure seems to indicate three different phases, each characterized by a different learning rate. Phase I, from time step 0 to 750 (10 in the graphic's scale), shows the steepest slope. In phase II, from 750 to about 4,500, learning accrues at a much slower yet steady rate. Beyond this point, some oscillation occurs; nonetheless, the system manages to improve a little further, reaching by the end of the run a maximum performance of approximately 114.

How good is this level of performance? In general, the optimal level of performance equals the weighted average of the expected value of the reward Y at the best possible classifiers, where the weights are of course the corresponding frequencies. The best possible classifiers are those whose schemata coincide with the conceptual schemata h_j and their predictions yield (on average) the highest Y. In M11, all concepts occur equally likely and have essentially the same generating distribution, so the optimal level of performance boils down to .96 times the expected value of Y at the best possible prediction (simply ignoring the small contribution due to successes obtained by sheer chance when predicting noisy observations). What is the best possible prediction? It turns out that the expected value of Y is not a monotone function of scope, so the best possible prediction includes sometimes as many 1's as possible and sometimes fewer; these bits are always located, of course, at the most likely spots under the corresponding

conceptual distributions. Since the difference between the maximum expected Y and the expectation at the widest prediction is relatively small (in the present case, these numbers are 123.1 and 119.6 respectively), the latter may be preferred for consistency and simplicity, and indeed it is used to compute the percentages given below.

Contrasting either figure with the observed 114 implies that the system is nearing the optimal performance level. Moreover, on average, the system finds 2.5 *correct* classifiers per run per concept (correct classifiers are those whose schemata match one of the conceptual schemata; correct classifiers nearly always exhibit a correct prediction). But not all concepts are typically covered by correct classifiers: more general classifiers attending each to more than one concept are found as well. A peculiar feature is that typically only a small fraction (about 20%) of the population is 1,000 observations old or older; out of those, many are less than 500 observations old. Thus, the population undergoes constant renovation, and relatively few classifiers survive the heavy competition and intensive pruning. However, this phenomenon does not seem to interfere with performance, which never experiences severe reverses. It also suggests the convenience of an annealing schedule where the exploration rate is decreased as learning accrues.

We know briefly discuss a few variations of M11 where rather satisfactory results were obtained; each of the following figures was obtained from a set of four independent runs of 4,000 trials each. Performance maintains above 80% of the "optimal" level when the specificity of conceptual schemata is decreased uniformly (from 4/11 to 4/17) and when the noise rate is increased (from 4% to 20%). Performance remains above 70% when regularities are blurred -- using Beta(3,1) distributions instead of Beta(5,1) -- and even when stimuli themselves are contaminated by flipping at random coordinates from h_i at a rate of 20%. These results indicate that PASS is able to detect useful patterns under relatively demanding conditions.

PASS can also learn environments whose concepts exhibit different frequency rates (provided these are not very low). A virtually error-free level is achieved when regularities are characterized by sharper Beta(11,1) distributions; these regularities are naturally found faster. On the negative side, the system seems to have trouble picking up low-frequency concepts (as resources tend to be taken up by high-frequency concepts), and can not describe bimodal regularities (as eg. those induced by Beta (1/2,1/2) distributions) even under ecological reinforcement (only the most likely modes are maintained in each case). These problems remain open for

future research.

As regards system parameters, here are some preliminary insights based on experience with M11. A moderate penalty seems useful, particularly during early stages of learning. It may not be necessary to let specificity affect the bidding/auction process (a controversial issue in the classifier system literature). It appears that the contribution to learning of `intersect` or `GA` is negligible relative to that of `explain`. In fact, the problem is similar for both procedures: whenever several concepts point to the same regions of the unit interval (as in M11), the restricted mating policy in PASS is insufficient to prevent fruitless mixing (see sections 2 and 5). As mentioned earlier, a change in mating policy seems appropriate. Finally, as an alternative to keeping a fixed population size, we have achieved some success by dynamically reducing it (along with the number of winners). This was done manually, although it could be obviously automated by looking at trends in the learning curve. The idea seems new in the classifier system framework and opens an interesting avenue in the direction of self-organization (see eg. [Fritzke 1993]).

PASS was also compared to the tree-oriented, continuous response algorithm (CON)FIRM, [Hawkins 1990]. Since FIRM proceeds in batch mode, this comparison prompts the study of resampling in PASS. Specifically, training samples of size 500 drawn from a "difficult" environment (not shown here) with 12 concepts and widely different u_i are made available to both FIRM and PASS, and trees output by FIRM are confronted with the resulting populations of classifiers after 4,000 iterations. A PASS-like prediction is generated for each terminal node on the basis of the corresponding mean and standard deviation from the training sample. Performance is then analyzed in terms of a test sample of 1,500 fresh observations from the same environment. To give an idea of the complexity of the task, the FIRM trees have an average of 23 terminal nodes, while PASS is restricted to 50 classifiers.

Results are encouraging. In general, PASS does not perform much worse than FIRM, and sometimes it does just as well. This is meritorious, for PASS never has access to all data simultaneously. Further, it is not hard to devise simple environments where FIRM's one-predictor-at-a-time strategy fails to the point of quitting at the root node! In contrast, PASS is able to learn such environments, although it takes some time to do so. These conclusions are in general agreement with those reached in [Quinlan 1988] (see also [Booker 1989]).

We conclude this quick review by examining a second type of data source. We are now interested in evaluating the system's ability to work with continuous predictors and to represent high-level models as eg. linear regression models. To this end, consider a simple regression model $y=x+\varepsilon$, where $0<x<1$ and ε follows a $N(0,\sigma^2)$ distribution. To transform x into a boolean vector, gray coding has better properties than binary coding, cf. [Caruana and Schaffer 1988]. Samples of 200 observations generated by this model -- with x uniformly distributed over the unit interval, six bits of precision in gray code and σ not greater than .1 -- were made available to PASS. After 2,000 iterations, a version of the system was able to reconstruct the model and reach a 90% performance level using only 25 classifiers. The only apparent requirement is that the auction be conservative ($\alpha=2$, $\beta=0$). Table 3 shows a typical set of solution classifiers and illustrates the kind of output provided by PASS; the columns are, respectively, s, d, S, age, estimated probability of match, type (and history) of classifier and f (only nonzero probabilities rounded to two digits are written).

Table 3 First 15 classifiers for simple regression

```
 0 1001?? 0000000000000111 2508.7  160   9 S 6 71 24
 1 ?100?? 0000000111000000 2499.0 1225  13 E 41 46 13
 2 000??? 1110000000000000 2390.1  108  14 E 44 44 11
 3 00??1? 1111100000000000 2199.8  519  15 S 3 51 19 24 3
 4 10?0?? 0000000000000111 2081.7  894  14 E 25 20 55
 5 010??? 0000001110000000 1909.7 1227  12 E 37 45 18
 6 0?1??? 0111110000000000 1895.8  407  24 XN 15 13 20 22 29
 7 00???0 1111100000000000 1836.9  175  13 A1 7 44 20 25 5
 8 010??? 0000011100000000 1826.4 1222  12 E 8 42 50
 9 1?1??? 0000000000111110 1717.1 1985  26 E 26 29 19 15 10
10 01?1?? 0000111100000000 1647.8  906  12 S 1 30 46 22
11 11???? 0000000111110000 1588.7  727  26 EF 13 16 16 32 23
12 0??0?? 0111110000000000 1561.1 1887  24 RN 19 12 20 23 27
13 0001?? 1111000000000000 1471.2 1308   6 S 28 56 15 1
14 1?1??? 0000000000111110 1283.3 1227  27 IF 25 31 19 16 9
15 0?0?1? 1111100000000000 1232.7  175  10 A1 4 62 13 21 0
```

Schemata exhibit a reasonable level of generality and predictions cover the entire unit interval homogeneously. Similar results were obtained using other kinds of linear models as, for example, 2^k factorial models.

7 Concluding remarks

This article has presented a new approach to automated data analysis based on the classifier system (CS) framework. A system implementing the new approach, called PASS (Predictive Adaptive Sequential System), has

been described and shown to solve satisfactorily a variety of instances of a particular regression problem. PASS also fares reasonably well against a batch-processing system and is capable of expressing high-level models as certain types of linear regression models. Overall, the approach herein discussed provides a flexible and robust new tool for data analysis. In addition, the new approach seems versatile enough to easily accommodate various enhancements from other areas in machine learning or statistics.

On the other hand, PASS inherits some of the pending problems in CSs and also introduces some new ones. The role of certain system parameters needs to be further examined, and some insights should be provided on scale-up factors, particularly those concerning rule-generation mechanisms. Extensions to deal with categorical, multivariate or time-dependent data should be addressed as well. Finally, a careful evaluation of PASS with respect to the bayesian approach seems of foremost interest. Indeed, one may want to investigate also the possibility of cooperation between both approaches. The conjecture is that bayesian ideas may have an important role to play in general CSs. The data-analytic problem introduced by PASS constitutes a natural context to begin to ascertain the nature of this role.

References

[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.

[Caruana and Schaffer 1988]
R. A. Caruana and J. D. Schaffer. *Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms.* In J. Laird (Ed.) Proceedings of the Fifth International Conference on Machine Learning. 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.

[Davis and Steenstrup 1987]
L. Davis and M. Steenstrup. *Genetic Algorithms and Simulated Annealing: An Overview.* In L. Davis (Ed.) Genetic Algorithms and Simulated Annealing. Morgan Kauffman, Los Altos, CA.

[Fritzke 1993]
B. Fritzke. *Growing Cell Structures -- A Self-organizing Network for Unsupervised and Supervised Learning.* Technical Report 93-026, International Computer Science Institute, Berkeley, CA.

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

[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.

[Hawkins 1990]
D. M. Hawkins. *Formal Inference-based Recursive Modeling.* Technical Report # 542, University of Minnesota, Minneapolis, MN.

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

[Holland 1986]
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 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 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.

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

[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.

[Quinlan 1988]
J. R. Quinlan. *An empirical comparison of genetic and decision-tree classifiers*. In J. Laird (Ed.) Proceedings of the Fifth International Conference on Machine Learning, Morgan Kauffman, San Mateo, CA.

[Riolo 1987]
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 1989]
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.

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

[Tierney 1990]
L. Tierney. *LISP-STAT. An Object-Oriented Environment for Statistical Computing and Dynamic Graphics*. John Wiley, New York.

[Titterington et al. 1985]
D. M. Titterington, A. F. M. Smith and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley, New York.

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

[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.