Speeding-up Offline-Learning Learning Classifier Systems: Modeling Time and Accuracy

Jaume Bacardit[1], David E. Goldberg[2], Martin Butz[2], Xavier Llora[2],Josep M. Garrell[1]

[1] Intelligent Systems Research Group, Universitat Ramon Llull, Psg. Bonanova 8, 08022 Barcelona, Catalonia, Spain, Europe.
{jbacardit,josepmg}@salleURL.edu
[2] Illinois Genetic Algorithms Laboratory (IlliGAL), Department of General Engineering, University of Illinois at Urbana-Champaign, 104 S. Mathews Ave, Urbana, IL 61801.
{deg,butz,xllora}@illigal.ge.uiuc.edu

Abstract. Windowing methods are useful techniques to reduce the computational cost of Pittsburgh-style Genetic-Based Machine Learning techniques. If used properly, they additionally can be used to improve the classification accuracy of the system. In this paper we develop a theoretical framework for a windowing scheme called *ILAS*, developed previously by the authors. The framework allows us to approximate the degree of windowing we can apply to a given dataset as well as the gain in run-time. The framework sets the first stage for the development of a larger methodology with several types of learning strategies in which we can apply *ILAS*.

1 Introduction

The application of genetic algorithms (GA) [1, 2] to classification problems is usually known as genetic based machine learning (GBML). One of the traditional ways of addressing it has been the Pittsburgh approach, early exemplified by LS-1 [3]. Usually systems applying this approach have a high computational cost, because each fitness computation means classifying the whole training set.

We can primarily reduce this cost by either: (a) decreasing complexity of the individuals or (b) decreasing fitness evaluation time. The former methods are usually referred to as *parsimony* methods [4]. The latter methods are either characterized as feature selection methods, reducing the number of problem attributes, or as incremental learning or windowing methods, reducing the number of training instances per fitness evaluation.

In previous work [5], we empirically tested some training set reduction schemes. These schemes select a training subset to be used for fitness computation. Changing the subsets at every iteration of the *GA* process, the scheme is a kind of windowing method. Our previous results showed that the techniques achieved the run-time reduction objective with small or nonexistent accuracy loss. Sometimes, test accuracy actually increased, indicating knowledge generalization effects that may alleviate over-fitting.

Several open questions remained. From a run-time reduction perspective, we are interested in deriving a model of the maximal learning time reduction we can achieve while avoiding significant accuracy loss. From a learning perspective, we are interested in the learning time reduction that maximizes accuracy in the system, given a constant run time. In order to achieve the latter objective, we need to develop a run-time cost model.

This paper addresses these points. Concentrating our efforts on only one of our windowing schemes, called *ILAS* (incremental learning with alternating strata), we first analyze when a problem gets difficult for *ILAS*. Once we answer this question, we develop a cost model of the system. With the two elements in hand, we finally construct a theory that provides an estimate for optimizing run-time as well as accuracy performance of the system.

The paper is structured as follows: Section 2 presents some related work. Then, we describe the framework of our classifier system in section 3. Section 4 describes the *ILAS* windowing scheme and some previous results that show the motivation of this paper. Section 5 contains the theoretical models of *ILAS* presented in this paper and some discussion. Finally, section 6 discusses the conclusions and some further work.

2 Related work

All run-time reduction methods related to training examples share a common idea: using a subset of the training examples for the learning process. From a general Machine Learning point of view, we can distinguish three main categories:

- **Wrapper Methods** [6]. These methods interactively select most suitable examples for the learning process. The subset of training examples used varies through the iterations until a stopping criteria is met. The criteria usually is based on the estimation that the current subset of examples is similar enough to the whole set.
- **Modified learning algorithms** [7]. These algorithms either are able to learn incrementally from subsets of the training instances, or they include and discard instances based on knowledge-representation specific information.
- **Prototype Selection** [8]. These methods apply a preprocessing stage in which the training set is reduced before the actual learning process. Unlike the two previous categories, prototype selection does not interact with the learner.

The *ILAS* windowing scheme studied in this paper belongs to the second of the categories described above.

3 Framework

In this section we describe the main features of our classifier system. GAssist (*Genetic Algorithms based claSSIfier sySTem*) [9] is a Pittsburgh-style classifier system based on GABIL [6]. Directly from GABIL we have taken the knowledge representation for discrete attributes (rules with conjunctive normal form (CNF) predicates) and the semantically correct crossover operator.

Matching strategy: The matching process follows a "if ... then ... else if ... then..." structure, usually called a *decision list* [10].

Control of the individual's length: Dealing with variable-length individuals raises some important issues. One of the most important one is the control of the size of the evolving individuals [4]. This control is achieved in GAssist using two different operators: (1) *Rule deletion*. This operator deletes the rules of the individuals that do not match any training example. This rule deletion is done after the fitness computation and has two constraints: (a) the process is only activated after a predefined number of iterations (to prevent a massive diversity loss) and (b) the number of rules of an individual never goes below a threshold. This threshold is assigned to the number of classes of the domain. (2) *Minimum description length-based fitness function*. The Minimum Description Length (*MDL*) principle is a metric applied in general to a theory (being a rule set in this paper) which balances the complexity and accuracy of the rule set. In previous work we developed a fitness function based on this principle. A detailed explanation of the fitness function can be found in [11].

4 The *ILAS* Windowing Scheme

In this section we describe the windowing scheme we are using in this paper. We also include some previous results motivating the research in this paper.

4.1 Incremental Learning with Alternating Strata

The *ILAS* scheme is basically a standard Pitt-style *GBML* system in which the training set has been stratified into s subsets of equal size. Each strata maintains approximately the class distribution of the whole training set. Each GA iteration utilizes a different strata to perform its fitness computation, using a round-robin policy. Figure 1 contains the pseudocode of *ILAS*.

4.2 Previous results and motivation

Figure 2 show results from [5] of the *ILAS* scheme applied to some datasets (Wisconsin Breast Cancer (bre) , Ionosphere (int), Pima-indians-diabetes (pim), Pen-Based Recognition of Handwritten Digits (pen), SatImage (sat), Thyroid Disease (thy)) from the University of California at Irvine (UCI) repository [12]. The first three datasets are small (less than 1000 instances), while the rest of datasets are of medium size (ranging from 6435 to 10992 instances). For the

Fig. 1. Pseudocode of the Incremental Learning with Alternating Strata scheme

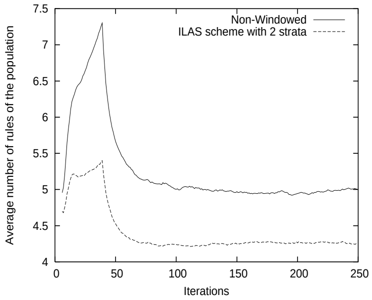
```
Procedure Incremental Learning with Alternating
            Strata
Input : Examples, NumStrata, NumIterations
Initialize GA
Reorder Examples in NumStrata parts of approximately
equal class distribution
Iteration = 0
SegmentSize = size(Examples)/NumStrata
While  Iteration < NumIterations
    If Iteration = NumIterations − 1 Then
        TrainSet = Examples
    Else
        CurrentSegment =Iteration mod
                          NumStrata
        TrainSet= examples from
            Examples[CurrentSegment·
                      SegmentSize] to
            Examples[(CurrentSegment + 1)·
                      SegmentSize]
    EndIf
    Run one iteration of the GA with TrainSet
    Iteration = Iteration + 1
EndWhile
Output : Best individual (set of rules) from GA population
```

small datasets we tested *ILAS* using $s = 2$, $s = 3$ and $s = 4$ strata and for the medium datasets we used 5, 10 and 20 strata.

The *ILAS* scheme is compared to the standard non-windowed system, labeled *NON*. The table includes results for accuracy and speedup (time of the original system over time of the windowed system, using the same number of iterations). Note that some speedup values are larger than the expected value of $1/s$. The cause is an implicit generalization pressure introduced by the windowing producing smaller individuals, which are faster to evaluate. This fact is also shown in figure 2 for the Pima dataset.

Fig. 2. Previous results of ILAS and plot of individual size reduction. Dat=dataset, Sch = windowing scheme, Acc=Test accuracy, Spe=Speedup

Dat	Sch	Acc	Spe	Dat	Sch	Acc	Spe
bre	NON	95.6%	—	pen	NON	79.9%	—
	ILAS2	95.9%	2.72		ILAS5	79.9%	5.18
	ILAS3	96.0%	4.63		ILAS10	79.4%	10.37
	ILAS4	95.8%	5.70		ILAS20	78.9%	20.44
ion	NON	89.5%	—	sat	NON	79.9%	—
	ILAS2	90.2%	2.72		ILAS5	79.9%	4.73
	ILAS3	90.6%	4.63		ILAS10	79.4%	9.04
	ILAS4	91.0%	5.70		ILAS20	78.9%	16.54
pim	NON	75.2%	—	thy	NON	93.6%	—
	ILAS2	74.8%	2.67		ILAS5	93.7%	5.20
	ILAS3	74.6%	4.41		ILAS10	93.6%	9.84
	ILAS4	74.0%	5.85		ILAS20	93.5%	18.52



The datasets shown in figure 2 exhibit different behavior patterns. The runs in the small datasets show that accuracy increases in *bre* and *ion* when using *ILAS* but not in *pim*. Moreover, the maximum accuracy for *bre* is achieved using 3 strata, while in *ion* it is achieved using 4 strata. In the large datasets, a larger number of strata slightly decreases accuracy while strongly improving computational cost. Thus, using *ILAS* can be beneficial in two aspects: an actual accuracy increase may be achieved in small datasets; strong run-time reduction is achieved, while only slightly decreasing accuracy. We are interested in how *ILAS* may be applied to achieve optimal results focusing on learning time and learning accuracy with respect to the number of strata s.

In the next section, we first develop a model of what makes a dataset hard for *ILAS*. Once we achieve this objective and we know which is the maximum number of strata we can use for a dataset, we can decide with how many strata *ILAS* should be applied to a given problem.

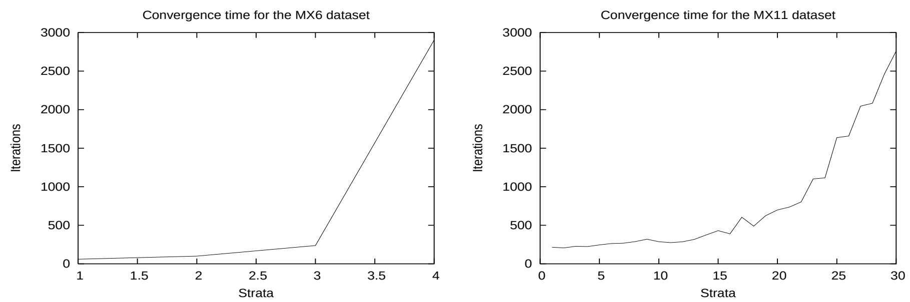
5 Analysis of the *ILAS* windowing scheme

This section presents our models for the hardness of a dataset for *ILAS* and a computational cost model. The models are crucial for estimating the optimal ILAS settings for a given problem.

5.1 What makes a problem hard to solve for ILAS?

We start our study focusing on the multiplexer [13] family of problems—a widely used kind of datasets with a well-known model. Our first step is to perform experiments determining how many iterations are needed to achieve 100% accuracy (convergence time) using the *ILAS* scheme for a given number of strata. The results of the experiments for the 6 (MX6) and 11 (MX11) bits multiplexer are shown in Figure 3. The plots are averaged over 50 independent runs. [3]

Fig. 3. Convergence time for the MX6 and MX11 datasets



For both datasets we can see that the convergence time increases with the number of strata in an exponential way. Before a certain break point, the first

[3] Unless noted otherwise, parameters were set as follows: Crossover probability 0.6; tournament selection; tournament size 3; population size 300; probability of mutating an individual 0.6; initial number of rules per individual 20; probability of "1" in initialization 0.75; Rule Deletion Operator: Iteraction of activation: 5; minimum number of rules: number of classes of domain +3; MDL-based fitness function: Iteration of activation 25; initial theory length ratio: 0.075; weight relax factor: 0.9.

part of the curve can be approximated by a linear increase. This break point is the maximum number of strata that is worth using in a dataset.

Intuitively we may suspect that after the break point the strata tend to miss-represent the whole training set causing learning disruptions. Since we know the optimal rule size in the multiplexer dataset, we are able to estimate how representative a strata may be. In the case of MX6 we have 8 rules, each rule covering 8 instances. In the case of MX11 we have 16 rules, each one covering 128 instances. Only by observing these numbers it is quite easy to see that MX6 has a higher risk of having one of these rules unrepresented in some strata, which translates into having a break point at strata 3 (as seen in figure 3).

In order to predict the break point, we calculate the probability of having a particular rule (which corresponds to a sub-solution) unrepresented in a certain strata. We can approximate this probability supposing uniform sampling with replacement:

$$P(\text{unrepresented rule}/s) = (1-p)^{\frac{D}{s}} \tag{1}$$

where p denotes the probability that a random problem instance represents a particular rule, D is the number of instances in the dataset and s is the number of strata. The probability essentially estimates the probability that a particular rule is not represented by any problem instance in a strata.

A general probability of success (requiring that no rule is unrepresented) of the whole stratification process can now be derived using the approximation $(1-\frac{r}{s})^s \approx e^{-r}$ twice to simplify:

$$P(success/s) = (1 - P(\text{unrepresented rule}/s))^{rs} \tag{2}$$

$$P(success/s) = e^{-rs\cdot e^{-\frac{pD}{s}}} \tag{3}$$

where r denotes the number of rules. The derivation assumes that p is equal for all rules which is the case for our experimental verification below. If p differs, a derivation of success is still possible but the closed form is not derivable anymore.

The model is experimentally verified for *MX6* and *MX11* in figure 4. The experimental plot is the average of performing 2500 stratification processes and monitoring when there was an unrepresented rule. We can observe that the theoretical model is quite close to the experimental data, although it is slightly more conservative.

If we overlap this probabilistic model with the convergence time curve we can see that the exponential area of the convergence time curve starts approximately when the success probability drops below 0.95. We show this observation in figure 5 for the MX6 and MX11 and also for two versions of MX6 that have 2 (MX6_2) and 4 (MX6_4) additional redundant bits, thus being more robust to the stratification process than MX6. We can approximately predict the break point, achieving one of the objectives of this paper.

5.2 Cost model of ILAS

The second objective of this paper is the development of a run-time model. Assuming constant run time per iteration, we can model the run-time of the

Fig. 4. Probability of stratification success. Verification of model with empirical data

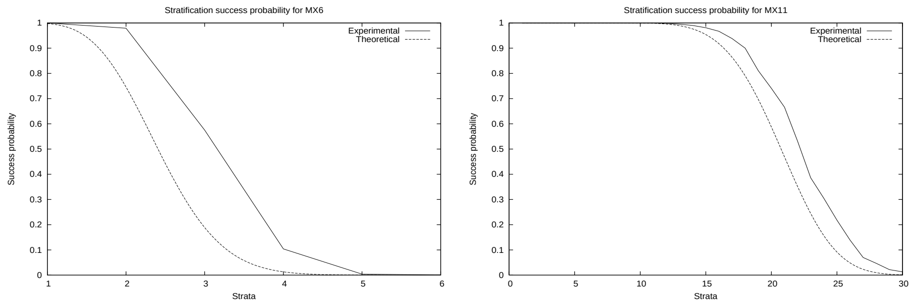
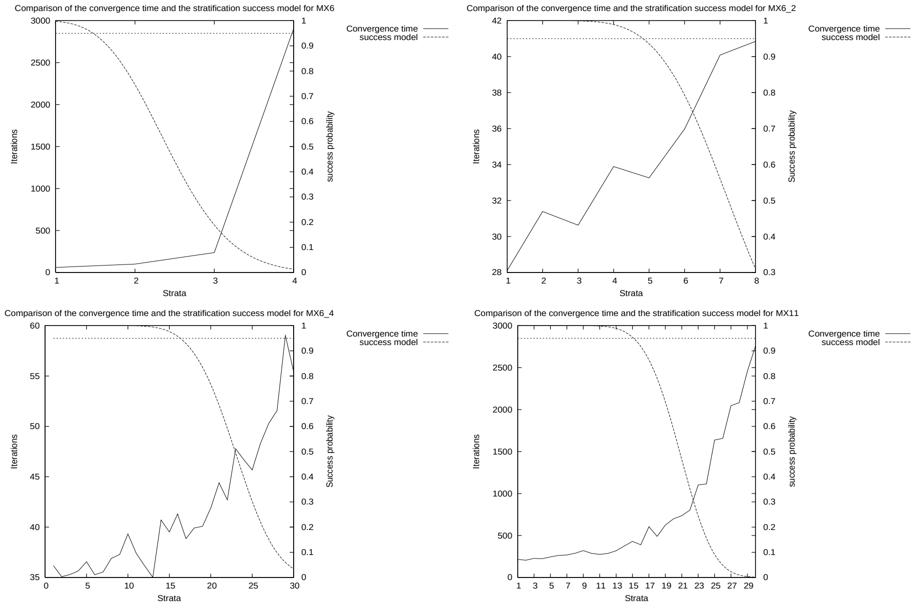


Fig. 5. Comparing the convergence time and the probability of stratification success. Vertical scale for left size of plots corresponds to iterations of convergence time. Scale for right side is probability of stratification success (equation 3)



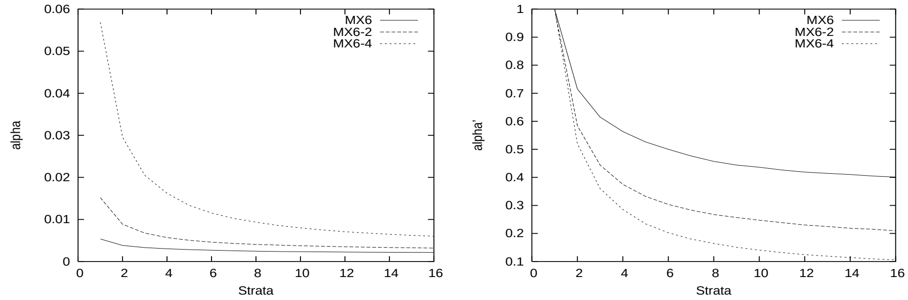
system by

$$T = \alpha \cdot it \tag{4}$$

where T denotes the total time of the learning process, α the time per iteration and it the number of iterations. Figure 6 shows α values for MX6,MX6_2 and MX6_4. Clearly, α is strongly dependent on the number of instances in a dataset. As hypothesized above, time approximately behaves inverse proportional to the number of strata. To have a better insight in α, we compute α' as $\alpha'_s = \alpha_s/\alpha_1$, that is, the value of α for s strata over the value for one strata. Figure 6 also shows α'.

The evolution of α' can be approximated by a formula such as $\alpha' = a/s + b$, where s is the number of strata and b is a constant that needs to be adjusted to the problem at hand (from applying the formula for 1 stratum we know that

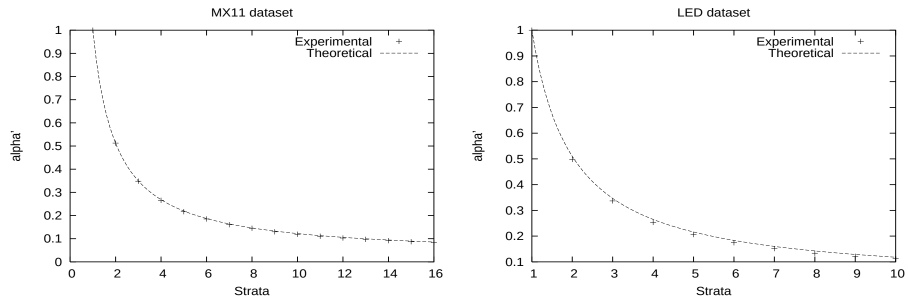
Fig. 6. α (time per iteration) and α' (α relative to a single stratum) values for some datasets



$a = 1 - b$). In order to assign a value to b effectively developing a predictive model for α', we did some tests with several datasets of the MX6 family (with redundant bits and redundant instances) and performed a regression process. The results showed that b is mostly correlated to the number of instances in the dataset, and can be modeled as $b = c/\#Instances + d$, applying regression again for c and d, which are dependent on the computer used for the tests.

The model of α' is verified experimentally with two different datasets: MX11 and an artificial dataset from the *UCI* [12] repository: LED (using 2000 instances with 10% of noise). The comparison of the model and the empirical data can be seen in figure 7, which shows that the model is quite accurate.

Fig. 7. Verification of the $alpha'$ model with MX11 and LED datasets



With this $alpha'$ model we can now deduce a formula to approximate the optimal number of iterations to maximize accuracy within a constant running time. The question is how many iterations (it_s) using s strata have the same run time as a base run time using one strata and it_1 iterations. it_s can be estimated by

$$it_s = \frac{it_1 \cdot s}{1 + b(s-1)}, \tag{5}$$

setting $a = 1 - b$. This formula is quite interpretable: b is the overhead of the *GA* cycle. If it were 0, the speedup obtained would be optimal and we could do as many iterations as $it_1 \cdot s$ for s strata. This overhead, however, also depends on the number of strata showing that the stratification does affect not only the evaluation stage of the *GA* cycle but also the resulting model.

6 Summary and Conclusions

This paper focused on a windowing scheme used originally to reduce the run-time of a Pittsburgh approach genetic-based machine learning system. Previous results suggested that the scheme could also improve the accuracy performance of the system. This paper showed how to extract a model to predict when this is possible.

In sum, we have developed two theories to model the behavior of the *ILAS* windowing scheme. The first one concerns the maximum number of strata that can be used to separate a dataset before the learning mechanism may be disrupted. The model is based on the number of rules that are needed to classify a dataset correctly and the number of instances that cover each rule. Our model is based on the probability of having all rules represented by at least one instance in each strata. The experimental validation confirmed the derived bound being slightly pessimistic about the outcome.

This model is based on the assumption that all rules represent an uniform number of instances. If the coverage of the rules is not uniform, the probability should decrease. However, given the slightly more conservative behavior of our model versus the empirical ratio of represented rules, we think that we can compensate, at some degree, having unbalanced rules. However, in order to have a fully usable model of *ILAS* we have to answer the pending question of how to model the quasi-linear increase in convergence time before the break point. Future research needs to address this issue in further detail.

The second theory developed in this section concerns the run-time of *ILAS*. Our model can predict the run-time reduction we can achieve in comparison to the system with 1 strata, given the supposition that all individuals (through all iterations) have the same rule size and rule distribution. Given datasets as *Pima*, shown in figure 2, we know that this is not necessarily true. However, the change in rule size is always decreasing when using *ILAS*. Consequently, our model may be considered as an upper bound in the general case.

With these two models we have constructed a framework that permits the use of the *ILAS* scheme in an efficient manner. Based on the expected complexity in the data, practitioners are now able to estimate the number of strata feasible to use in a dataset. They can also predict the relative run time reduction achievable. Future research should focus on putting the framework into practice on real problems, checking which kind of strategies for *ILAS* can be predicted successfully and reliably with these models.

Acknowledgments

The authors acknowledge the support provided by the Spanish Research Agency (CICYT) under grant numbers TIC2002-04160-C02-02 and TIC 2002-04036-C05-03, the support provided by the Department of Universities, Research and Information Society (DURSI) of the Autonomous Government of Catalonia under grants 2002SGR 00155 and 2001FI 00514.
Also, this work was sponsored by the Air Force Office of Scientific Research, Air

Force Materiel Command, USAF, under grant F49620-03-1-0129, and by the Technology Research, Education, and Commercialization Center (TRECC), at University of Illinois at Urbana-Champaign, administered by the National Center for Supercomputing Applications (NCSA) and funded by the Office of Naval Research under grant N00014-01-1-0175. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the Technology Research, Education, and Commercialization Center, the Office of Naval Research, or the U.S. Government.

References

1. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
2. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Inc. (1989)
3. Smith, S.F.: Flexible learning of problem solving heuristics through adaptive search. In: Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Los Altos, CA, Morgan Kaufmann (1983) 421–425
4. Soule, T., Foster, J.A.: Effects of code growth and parsimony pressure on populations in genetic programming. Evolutionary Computation **6** (1998) 293–309
5. Bacardit, J., Garrell, J.M.: Comparison of training set reduction techniques for pittsburgh approach genetic classifier systems. In: Proceedings of the "X Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA2003)". (2003)
6. DeJong, K.A., Spears, W.M., Gordon, D.F.: Using genetic algorithms for concept learning. Machine Learning **13** (1993) 161–188
7. Fürnkranz, J.: Integrative windowing. Journal of Artificial Intelligence Research **8** (1998) 129–164
8. Salamó, M., Golobardes, E.: Hybrid deletion policies for case base maintenance. In: Proceedings of FLAIRS-2003. (2003) 150–154
9. Bacardit, J., Garrell, J.M.: Evolving multiple discretizations with adaptive intervals for a pittsburgh rule-based learning classifier system. In: Proceedings of the Genetic and Evolutionary Computation Conference - GECCO2003, LNCS 2724, Springer (2003) 1818–1831
10. Rivest, R.L.: Learning decision lists. Machine Learning **2** (1987) 229–246
11. Bacardit, J., Garrell, J.M.: Bloat control and generalization pressure using the minimum description length principle for a pittsburgh approach learning classifier system. In: Proceedings of the 6th International Workshop on Learning Classifier Systems, (in press), LNAI, Springer (2003)
12. Blake, C., Keogh, E., Merz, C.: UCI repository of machine learning databases (1998) (www.ics.uci.edu/mlearn/MLRepository.html).
13. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation **3** (1995) 149–175