A Mixed Discrete-Continuous Attribute List Representation for Large Scale Classification Domains

Jaume Bacardit
ASAP research group, School of Computer Science, Jubilee Campus, Nottingham, NG8 1BB. MyCIB, School of Biosciences, Sutton Bonington, LE12 5RD, University of Nottingham, UK
Jaume.Bacardit@Nottingham.ac.uk

Natalio Krasnogor
ASAP research group, School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, UK
Natalio.Krasnogor@Nottingham.ac.uk

ABSTRACT

Datasets with a large number of attributes are a difficult challenge for evolutionary learning techniques. The recently proposed *attribute list rule representation* has shown to be able to significantly improve the overall performance (e.g. run-time, accuracy, rule set size) of the BioHEL Iterative Evolutionary Rule Learning system. In this paper we, first, extend the attribute list rule representation so it can handle not only continuous domains, but also datasets with a very large number of mixed discrete-continuous attributes. Secondly, we benchmark the new representation with a diverse set of large-scale datasets and, third, we compare the new algorithms with several well-known machine learning methods. The experimental results we describe in the paper show that the new representation is equal or better than the state-of-the-art in evolutionary rule representations both in terms of the accuracy obtained with the benchmark datasets used, as well as in terms of the computational time requirements needed to achieve these improved accuracies. The new attribute list representation puts BioHEL on an equal footing with other well-established machine learning techniques in terms of accuracy. In the paper, we also analyse and discuss the current weaknesses behind the current representation and indicate potential avenues for correcting them.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Concept Learning, Induction*; J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and Genetics*

General Terms

Algorithms, Experimentation, Performance

Keywords

Evolutionary Algorithms, Learning Classifier Systems, Rule Induction, Large-scale Datasets

GECCO'09, July 8–12, 2009, Montréal, Québec, Canada.
Copyright 2009 ACM 978-1-60558-325-9/09/07 ...$5.00.

1 Introduction

Learning Classifier Systems (LCS) [16] and, in general, evolutionary learning techniques [14], have experienced remarkable progress over the past few years. This progress has been related to a multi-pronged research programme that included vigorous research on, e.g., theoretical models [8], advanced exploration mechanisms [8], new representations [17], the use of local search [3] and a healthy mix of "reality checks" against hard classification problems [19].

The application of evolutionary learning techniques to mine large-scale datasets has received renewed emphasis and systematic studies on this research front are being carried out. Two recent examples in this line of inquiry are NAX [17] and BioHEL (Bioinformatics-oriented Hierarchical Evolutionary Learning) [4]. These two learning systems apply the iterative rule learning approach first proposed by Venturini [24]. Although great progress has been achieved for specific types of problems using the above methods (e.g. Bioinformatics domains [22, 5]), the important question of their generality and robustness across other domains remains unanswered. That is, notwithstanding The No Free Lunch theorems and related work [27, 21], it is important to assess how far it is possible to push the boundary of competency for a given algorithm, in this case, evolutionary learning ones.

Recently, a new rule encoding for continuous attributes, called attribute-list knowledge representation (ALKR), was proposed within the framework of BioHEL [2]. ALKR was designed to improve BioHEL's efficiency in datasets with a large number of attributes. Improved performance was achieved by identifying, throughout the iterations of the genetic algorithm, the key relevant attributes that contributed towards concrete classifications. That is, rules kept only information associated to useful attributes, thus avoiding unnecessary computations with attributes that did not contribute towards a classification decision. Experiments showed that ALKR helped not only to improve run-time but also to generate more accurate solutions. This was attributed to the fact that the search process could invest CPU effort only in the relevant attributes without investing in unimportant ones.

In this work our aim is three-fold. First, we extend the representation to be able to cope also with discrete attributes (using the GABIL [11] representation for these) and propose an efficient method to perform the matching process for datasets with both continuous and discrete attributes.

Secondly, we extensively evaluate the capacity of the representation (and, in general, the BioHEL system) to deal with large-scale datasets with different characteristics. Some of the datasets we use have a large number of instances (up to 500000 instances), others have a large number of attributes (up to 14000 variables) while a third type of datasets has unbalanced classes. The aim of this evaluation process is to do an initial assessment of the capacity of BioHEL with a mixed continuous-discrete ALKR as to identify strengths and weaknesses of our approach. Finally, we compare the performance of the representation against other standard machine learning techniques.

The rest of the paper is structured as follows: First, section 2 will describe some related work. Next, section 3 will describe the framework of this work: the BioHEL evolutionary learning system and the ALKR and GABIL representations. Section 3.2 will present our extension of the representation to handle both continuous and discrete attributes efficiently. Afterwards section 5 will describe the experimental design, section 6 will show and analyze the results of our experiments and, finally, section 7 will contain the conclusions and further work of the paper.

2 Related work

Several methods have been proposed through the years to alleviate the run-time of LCS and evolutionary learning methods. Rule representations [18, 17] that use SSE or Altivec vectorial instructions to parallelize the match operations have been proposed, by performing various match operations (at the attribute level) at the same time. A different approach, with a similar inspiration as the rule representation studied in this paper was proposed by Butz et al.[9]. This paper studied a specificity-based rule representation, where the attributes were reordered from specific to general, and the match process started with the most specific ones. Experiments showed that this representation was more efficient than traditional rule representations. The set of relevant attributes identified by ALKR are usually the most specific attributes in the domain, so effectively both representations achieve higher efficiency by performing a similar procedure.

The maintenance of a relevant attribute list effectively represents a feature selection (FS) [15] strategy. There are various works on the use of evolutionary computation for FS (e.g. [23]). Most of these works define a binary chromosome with one bit associated to each attribute of the domain, which decides whether the feature is selected or not. Fitness functions of these methods try to reduce the subset of selected features while attempting to preserver a high accuracy. ALKR applies an slightly different approach for two reasons. (1) feature selection and learning are performed in a concurrent and integrated manner, and (2) our feature selection is performed at a *rule-wise* level. That is, different rules can use different sets of relevant attributes. Standard FS methods filter the dataset keeping only the same subset of attributes for the whole solution (e.g. rule set). Thus, we could say that our representation applies a *fine-grained* FS process.

Finally, we would like to mention another system, Evolutionary Concept Learner (ECL) [13] that uses an evolutionary approach to inductive logic programming. Rules are Horn Clauses having a list of terms, each of them associated to an attribute. Terms can be added or removed to/from the list using mutation operators, which is similar to the knowledge representation studied in this paper. However, ECL does not have a crossover operator nor was not designed for efficiency purposes.

3 Framework

In this framework section we would like to describe the learning system that has been used for our experiments (BioHEL) the studied representation, ALKR, and the GABIL representation [11] that we will use for the discrete attributes.

3.1 The BioHEL evolutionary learning system

BioHEL, originally developed for structural bioinformatics datasets, is an evolutionary learning system following the Iterative Rule Learning approach, first used in evolutionary learning by Venturini [24]. Most of BioHEL's modules have been adapted from GAssist [1] which is a Pittsburgh LCS. The system applies a standard generational GA, which evolves individuals that are classification rules.

The final solution of the learning process is a rule set that is constructed by applying iteratively a GA. After each rule is obtained, the training examples that are covered by the rule are removed from the training set, to force the GA of the next iteration to explore other areas of the search space. The rules are inserted into a rule set with an explicit default rule that covers the majority class of the domain. The evolved rules will cover all the other classes. The iterative process of generating rules stops when it is impossible to find any rule where the associated class is not the majority class of the matched examples. When this happens, all remained examples are assigned to the default rule. Also, several repetitions of the GA with the same set of instances but different random seeds are performed. A new rule is inserted into the rule set (and therefore examples removed from the training set) only when it is the best rule obtained from across all the incumbent GA runs.

The system uses a windowing scheme called ILAS (incremental learning with alternating strata) [1] to reduce the run-time of the system. This is especially useful for dataset with a high number of instances such as the ones studied in this paper. This mechanism divides the training set into several non-overlapping subsets and selects a different subset at each GA iteration for the fitness computations of the population.

The fitness function of BioHEL is based on the Minimum Description Length (MDL) principle [20]. The MDL principle is a metric applied to a theory (a rule) which balances its complexity and accuracy. For the specific details of the overall MDL formula and the complexity definition for the GABIL representation, please see [4]. The details of the complexity definition for the continuous attributes ALKR version are in [2].

As to improve accuracy and leveraging the fact that BioHEL is a stochastic algorithm, several rules sets are generated by running it multiple times with different random seeds. These rule sets are ensembled based on a simple majority vote, thus combining their predictions. In the rest of the paper, all reported accuracy measures of BioHEL will be the result of a such an ensemble prediction.

3.2 The Attribute List Knowledge Representation (ALKR)

We describe next the ALKR [2] as it forms the core of our enhanced mixed continuous-discrete attribute list representation.

3.2.1 *Representation definition*

Each rule will be represented by four elements: (1) an integer containing the number of expressed attributes, (2) a vector specifying which attributes are expressed, ordered in the same way as they are defined in the dataset, (3) a vector specifying, for each expressed attribute, the lower and upper bound of its associated interval and (4) the class associated to the rule.

3.2.2 *Initialization strategy*

The initialization of the rules in any kind of evolutionary learning system is crucial for a proper learning process, and it becomes critical when dealing with problem domains that contain hundreds of attributes. The proper balance between specificity and generality needs to be found. This issue has been studied in depth for Michigan LCS [8]. For continuous problems, a rule representation would need to decide about two specific issues: (1) how many and which attributes would be relevant within a rule and (2) how would intervals be defined for the relevant attributes.

As suggested by the representation of the NAX system [17], this representation addresses the first issue by having a parameter that specifies the expected value of the number of relevant attributes in a rule. The expected value is transformed into a probability and then, using this probability, the relevant attributes are randomly chosen. The answer for the second question means having to decide the size of the interval and its position within the domain of the attribute. The size of the interval is randomly initialized with uniform distribution to be between 25% and 75% of the domain size. Moreover, a covering mechanism similar to [25] is used. This mechanism samples without replacement and with class-wise uniform probability an example from the training set and initializes the interval to be centered at the value of that instance for the attribute being initialized. In case the interval would overlap with lower or upper bound of the domain, it is shifted to place it fully within the attribute domain.

3.2.3 *Exploration operators*

This representation's search space is explored by means of two types of operators: (1) the traditional crossover and mutation operators that edit the intervals for the expressed attributes and (2) the operators that edit the list of expressed attributes.

The crossover operator of this representation could be defined as a "virtual one point crossover". In a standard hyperrectangle representation, a one point crossover operator would randomly select a cut point within the chromosome laying in the same position (attribute) for both parents and exchange all the genes after the cut point between parents. This procedure follows these steps:

1. An expressed attribute from the first parent is randomly chosen
2. If the attribute is also expressed in the other parent then
   (a) A cut point is selected, either before the lower bound, between the two bounds or after the upper bound, and the genes are exchanged accordingly between the parents
   (b) The intervals for the rest of attributes in the list are exchanged between parents
3. If the attribute is not expressed in the second parent then
   (a) The next attribute (in the order in which the attributes are defined in the dataset) that is expressed in the second parent is identified
   (b) The complete intervals (without mixing bounds) after the cut point are exchanged between parents: Parent 1 gives all the intervals after the cut point attribute and parent 2 gives the intervals starting at the next expressed attribute after the cut point attribute
4. One offspring randomly picks the class value from one parent and the other offspring from the other parent.

The mutation operator selects one bound with uniform probability and adds or subtracts a randomly generated offset to the bound, of size (picked with uniform distribution) between 0 and 50% of the attribute domain. If the mutation affects the class value of the rule, a different class value is assigned to the rule, picked at random. Both crossover and mutation operators can create intervals where the lower bound is higher than the upper bound. In this case the rules are repaired by swapping the bounds.

The *specializing* and *generalizing* operators are used to respectively add and remove attributes to/from the list of expressed attributes. These operators are applied to the offspring population after mutation with a given individual-wise probability. When the generalizing operator is applied to an individual, one of the expressed attributes is randomly selected with uniform probability and afterwards the attribute and its associated interval are removed from the rule. When the specializing operator is applied, one of the attributes from the domain not expressed in the rule is randomly selected and it is added to the list. An interval for the attribute is randomly initialized using the same policy explained above for the initialization stage.

3.2.4 *Match process*

The match process of a rule with this representation is simple:

- For each attribute in the list of expressed attributes:
  - Check if the instance's value for the attribute lays between the bounds
  - If not, the match process returns false, if true, continue with the next attribute
- Return true

3.3 The GABIL representation

The predicates in *GABIL*'s rules have a fixed structure: A Conjunctive Normal Form (CNF).

$$(A_1 = (V_1^1 \vee \ldots \vee V_1^m) \bigwedge \ldots \bigwedge A_n = (V_n^1 \vee \ldots \vee V_n^m))$$

Where A_i is the ith attribute of the problem and V_i^j is the jth value of the ith attribute.

This kind of predicate is encoded into a binary string in the following way: given a problem with two attributes, where each attribute can take three values {1,2,3}, a rule of the form "If the first attribute has value 1 or 2 and the second one has value 3 then we predict class 1" will be represented by the string 110|001||1. There is a bit associated to each value of each attribute. If a value appears in the disjunction associated to its attribute in the predicate, the bit is set to one. Otherwise it is set to zero. The | symbols are placed just for clarify, to differentiate the parts of the rule associated to each attribute and the associated class of the rule, but do not appear in the encoding.

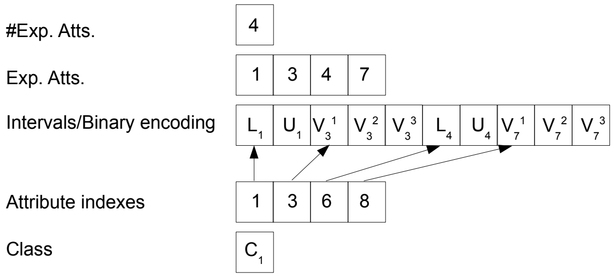
4 Extending ALKR to Deal with Both Continuous and Discrete Attributes

This section presents our extension to this knowledge representation so it can be applied to datasets that contain both discrete and continuous attributes. The GABIL representation [11] has been chosen to represent the discrete attributes. This extension is very simple and there is only one detail that needs to be designed with very care: the efficient matching process. The next paragraphs present the changes that have been done to the representation.

4.1 Representation definition

The only change in the representation definition is that instead of having a list of intervals, defined as a lower and an upper bound, we will have a mixed list of elements. Intervals will still be used for the continuous attributes, and the binary GABIL attribute encoding will be use for the discrete ones. Also, there is one addition: a list of attribute indexes that indicate, within the previous list, where the information associated to each attribute starts. This list is included for efficiency purposes. Figure 1 contains one example of this definition.

Figure 1: Example of a rule in the extended attribute list knowledge representation with four expressed attributes: 1, 3, 4, and 7. Attributes 1 and 4 are continuous, attributes 3 and 7 are discrete and have 3 values each. l_n = lower bound of attribute n, u_n= upper bound of attribute n, v_n^j = value j of attribute n, $c1$=Class 1 of the domain



4.2 Initialization strategy

The policy that selects which attributes are expressed in the initial rules based on a parameter encoding the expected value of expressed attributes remains unchanged. To initialize the GABIL binary encoding for the discrete attributes we define a probability of setting to one each of the bits associated to a value of the attribute. For each attribute, the covering-like operator sets to one the bit associated to the value from the seed example and initializes the rest of values with the probability of one. The list of attribute indexes is computed after the initial attributes are expressed.

4.3 Exploration operators

The crossover operator is almost unchanged. If the attribute selected as cut-point is expressed in both parents and is a discrete one, the actual cut-point for the recombination can be in any point within the GABIL binary encoding of the attribute. The list of indexes is regenerated for the offspring after crossover. Mutation in discrete attributes is the classic bit-flipping mutation. The generalize and specialize remain mostly unchanged, the only addition is that they regenerate the indexes list after their application.

4.4 Match and evaluation process

The match process is crucial in order to have an efficient representation. When thinking about a representation with a mixed kind of attributes, intuitively the first approach to perform the matching process would probably be like follows:

- For each attribute in the list of expressed attributes:
  - If the attribute is discrete:
    - Check if the bit associated to the instance's value is set to 1
    - If not, the match process returns false, if true, continue with the next attribute
  - Otherwise:
    - Check if the instance's value for the attribute lays between the bounds
    - If not, the match process returns false, if true, continue with the next attribute
- Return true

The main difference between this code and that from the original representation (in section 3.2.4) is the extra **If** operation performed *for each* expressed attributed. Considering that this code is the core of the whole learning system, where the program spends more time, this extra operation can produce a considerable impact in the efficiency of the system. To overcome this problem the algorithm precalculates which attributes are continuous and which are discrete. That is, although at each GA iteration, each rule in the population with all the training examples (or the current subset of examples, when using a windowing system such as ILAS) is evaluated, the algorithms needs only precalculate the lists of discrete and continuous attributes once at each iteration for each rule. The pseudocode for the whole evaluation process of a rule in a GA iteration is included in figure 2. This pseudocode assumes that the discrete values have been transformed into integers starting with 0.

5 Experimental design

This section contains the description of the experimental design of the results reported in this paper, including the different performed experiments and datasets, and the configurations of BioHEL.

Figure 2: Pseudo-code of the rule evaluation process with the extended ALKR

```
Procedure Rule Evaluation
Input : $rule$,$TrainingExamples$
$ListDiscrete$ = Compute list of discrete attributes in $rule$
$ListContinuous$ = Compute list of continuous attributes in $rule$
$fitnessAgent$ = Initialize fitness agent
Foreach $example$ in $TrainingExamples$
        $match = true$
        Foreach $attribute$ in $ListDiscrete$
                $value = example.values[attribute]$
                $index = rule.attributeIndexes[attribute]$
                If $rule.encoding[index + value] = 0$
                    $match = false$
                    Finish matching process for this example
                EndIf
        EndForeach
        Foreach $attribute$ in $ListContinuous$
                $value = example.values[attribute]$
                $index = rule.attributeIndexes[attribute]$
                If $values < rule.encoding[index]$
                    or $values > rule.encoding[index + 1]$
                    $match = false$
                    Finish matching process for this example
                EndIf
        EndForeach
        If $match$ is true
            If $rule.class = example.class$
                Add a correct classification to $fitnessAgent$
            Else
                Add a wrong classification to $fitnessAgent$
            EndIf
        Else
                Add a no-classification to $fitnessAgent$
        EndIf
EndForeach
$fitness$ = Compute rule fitness with $rule$ and $fitnessAgent$

Output : $fitness$
```

5.1 Experiments

In order to perform an analysis of the competence of ALKR on challenging large-scale datasets we have selected a set of seven datasets that contain a medium-high number of instances, variables or both. Table 1 contain the main characteristics of these datasets.

The adult (adu), kddcup, waveform (wav) and connect-4 (c-4) datasets are taken from the UCI repository [7] of machine learning datasets. The adult dataset contains data people extracted from the U.S. census bureau database, and the problem consists in predicting whether a given record belongs to somebody that earns more or less than 50K a year. The kddcup dataset is a classic large-scale benchmark. KDDCUP is a Data Mining and Knowledge Discovery competition organized by ACM SIGKDD, the Special Interest Group on Knowledge Discovery and Data Mining. Specifically, we are using the data from the 1999 edition of the KDDCUP, predicting intrusion detection. The waveform datasets is a synthetic problem that consists in recognizing the shape of a wave give 40 continuous characteristics. All the characteristics have been distorted with noise, and 19 of them are solely noise. The connect-4 dataset contains all possible moves in the connect-4 game where neither player has won yet nor the next move is forced. The prediction is the final outcome of the game.

The FARS (Fatality Analysis Reporting System) dataset is a compilation of statistics about car accidents made by the U.S. National Center for Statistics and Analysis [1]. The specific dataset used contains information about all people involved in car accidents in the U.S. during 2001. The selected class is the level of injury suffered. The Germinate (Germ) dataset contains data generated by microarray techniques from tissue extracted from seeds of the Arabidopsis Thaliana plant [6]. The class of the dataset is whether the seed germinates or not. Finally, the ParMX dataset is an hybrid synthetic parity-multiplexer dataset [8]. This dataset is a challenge for many machine learning techniques because the optimal solution consists in 512 (non-ordered) rules where half of the attributes of the domain are expressed in each rule.

All datasets have been partitioned into training and test sets using a stratified ten-fold cross-validation. All training and test partitions are publicly available in WEKA format at `http://www.infobiotic.net/datasets/gecco2009.tar.gz`. In these set of problems we have included some classic large-scale benchmarks such as the kddcup dataset as well as datasets with extremely high number of attributes as the germinate dataset.

To assess the competence of BioHEL using ALKR (labelled BioHEL-ALKR) we have compared it to other alternatives. First, we have also the previous version of BioHEL. This version uses either the NAX representation in the datasets that only have continuous attributes or a hybrid intervalar-GABIL representation without attribute list for the rest of datasets. This configuration is labelled as BioHEL-orig. Also, we have included in the comparison three well known machine learning datasets: C4.5, Naive Bayes and SVM. We have used the WEKA implementation [26] of the first two and the libsvm [10] implementation of the latter. Their default parameters have been used.

Table 1: Features of the datasets used in this paper. #Inst. = Number of Instances, #Attr. = Number of attributes, #Cont. = Number of continuous attributes, #Disc. = Number of discrete attributes, #Cla. = Number of classes, Dev.cla. = Deviation of class distribution

Name	#Inst.	#Attr.	#Cont.	#Disc.	#Cla.	Dev.cla.
adu	48842	14	6	8	2	26.07%
c-4	67557	42	0	42	3	23.79%
FARS	100968	29	5	24	8	13.08%
Germ	129	13942	13942	0	2	6.03%
kddcup	494020	41	26	15	23	12.89%
ParMX	262144	18	0	18	2	0%
wav	5000	40	40	0	3	0.36%

5.2 BioHEL configuration

For all the experiments BioHEL used the same configuration, summarised in table 2, except for two parameters: (1) the number of stata of the ILAS windowing scheme, which is set to 2 for datasets with small number of instances and to 50 otherwise and (2) the coverage breakpoint parameter of BioHEL's MDL-based fitness function. This parameter controls the generality pressure of the fitness formula. A value too high will generate over-general rules. A value too low will generate rules with small coverage and potentially suffering from over-learning. This parameter was adjusted for each dataset to be as high as possible while avoiding over-general rules. The values for these two parameters for all datasets are reported in table 3. As specified in table 2, 25 rule sets are used in each ensemble. This means that BioHEL is run 25 times on each training set with different

[1] Downloaded from `ftp://ftp.nhtsa.dot.gov/FARS/`

random seeds. This makes a total of 250 BioHEL runs per dataset and configuration (BioHEL-orig or BioHEL-ALKR).

Table 2: General parameters of BioHEL

Parameter	Value
General parameters	
Crossover prob.	0.6
Selection algorithm	tournament
Tournament size	4
Population size	500
Individual-wise mutation prob.	0.6
Default class policy	major
Iterations	50
Repetitions of rule learning process	2
Expected value of #expressed att. in init.	15
Probability of one in GABIL repr.	0.75
Rule sets per ensemble	25
MDL-based fitness function	
Iteration of activation	10
Initial theory length ratio	0.25
Weight relax factor	0.90
Coverage ratio	0.90
Attribute List Knowledge representation	
Expected value of expressed attributes	15
Prob. generalize	0.10
Prob. specialize	0.10

Table 3: Dataset-specific parameters of BioHEL

Dataset	#strata ILAS	Coverage breakpoint
adu	50	0.01
c-4	50	0.0025
FARS	50	0.1
germ	2	0.1
kddcup	50	0.1
ParMX	50	0.001
wav	2	0.1

6 Results

Table 4 describes the results of our experiments. For each dataset and learning system we report the cross-validation accuracy. For the two configurations of BioHEL as well as C4.5 we report the size of the generated solutions in terms of number of rules or number of leaves. For the two configurations of BioHEL we also report the average run-time and the average number of expressed attributes per rule. As these two configurations belong to the same code base, the run-times are directly comparable. We have not included run-times of the other systems because they are coded in various languages, so a totally fair comparison would not be possible. The accuracy results were statistically analyzed using the Friedman test for multiple comparisons (following [12]). The test indicated with a confidence of 97.77% that there were significant differences between the compared accuracies. A post-hoc Holm test indicated that BioHEL-ALKR was significantly better than Naive Bayes with 95% confidence.

If we compare the performance of the two BioHEL configurations we can see how BioHEL-ALKR is equal or better in terms of accuracy than BioHEL-orig in all but one of the datasets (*FARS*), which comfirms the observations from [2]. Our hypothesis for the higher performance of BioHEL-ALKR is that given that the rules only contain relevant attributes, the exploration operators are more efficient because they always take place at genes that are important for the rule. In BioHEL-orig it may happen often (especially in datasets such as *Germ*) that a crossover or mutation operator affects an irrelevant attribute, hence producing either no effect in most cases. We leave for further work the validation of this hypothesis.

In terms of complexity of the generated solutions, ALKR generated smaller rule sets in all but two of the datasets (*ParMX* and *wav*) and its rules had less number of expressed attributes per rule. In some cases, such as the *FARS* dataset, the ALKR rules had almost half the average number of relevant attributes of BioHEL-orig. The run time of BioHEL-ALKR was always shorter except for the *ParMX* dataset, one of the datasets with smaller number of attributes, and all of the same kind. Specially notable is the run-time difference in the *Germ* dataset, with almost 140000 attributes. ALKR was more than 72 times faster than the NAX representation, which was used for this dataset because it only contains continuous attributes. The design of ALKR, that only holds the relevant attributes of a rule, was fully exploited in this dataset where each rule only has an average 3.6 relevant attributes out of 14000.

BioHEL performs similarly or better than the other machine learning methods with the exception of the *wav* dataset, where it is outperformed by the SVM. This dataset is precisely the "least large-scale" dataset from those evaluated in this paper, so we can conclude that this method is well prepared to deal with large-scale datasets. Specially notable is the performance difference in the *Germ* dataset, where BioHEL is by far the best method. This kind of problem, with so many attributes and so few instances is very difficult to solve, and standard methods have a high danger of over-fitting the data because of the low number of instances. In these situations BioHEL benefits greatly from both ALKR and the ensemble mechanism wrapped over it. All the rule sets generated by BioHEL were totally different from each other. Their combination contained a large diversity of predictions that managed to increase the robustness of BioHEL. The rule sets generated by BioHEL were much more compact than C4.5's decision trees for all datasets except for *Germ* dataset.

Nevertheless, these experiments also show several places where BioHEL's performance can be improved. For instance, the number of rules that BioHEL generates for the ParMX dataset (402 rules) is much larger than the optimal solution of 257 (ordered) rules. BioHEL managed to generated accurate rules, but not the best ones. This indicates that the generality pressure of its fitness function can be improved. The value that we used for the *coverage breakpoint* parameter was the one that allowed the system to avoid generating over-general rules at the early stages of the learning process, but it produced sub-optimal rules in later iterations as we can see in figure 3. This figure plots, for each rule learnt by BioHEL, how many examples were removed from the training set. All the first rules are optimal[2], but afterwards the generality pressure effectively diminishes. Another situation where BioHEL's performance could be improved is the *c-4* dataset. This dataset has the second largest class unbalance degree. Looking at BioHEL's predictions, the minority class is very rarely predicted. Thus,

[2] Each optimal rule covers 512 examples of the *whole* dataset. As we are using cross-validation and each training set contains 90% of the dataset, the number of examples covered by optimal rules is smaller in our experiment

Table 4: Results of the experiments on large-scale datasets.

Dataset	Method	Accuracy	Size	#exp.att.	Run-time(s)
adu	BioHEL-orig	85.8±0.5	17.1±1.2	10.5±2.2	177.4±21.4
	BioHEL-ALKR	85.8±0.5	17.0±1.2	9.3±2.9	156.7±21.0
	C4.5	86.0±0.4	622.9±79.7	—	—
	Naive Bayes	84.1±0.5	—	—	—
	SVM	83.9±0.4	—	—	—
c-4	BioHEL-orig	80.7±0.4	113.1±7.8	19.3±4.5	2486.3±331.0
	BioHEL-ALKR	80.7±0.4	113.1±6.5	19.2±4.5	1914.4±248.4
	C4.5	80.9±0.5	4075.8±145.8	—	—
	Naive Bayes	72.1±0.2	—	—	—
	SVM	81.2±0.4	—	—	—
FARS	BioHEL-orig	79.9±0.3	124.8±9.1	20.5±2.8	3181.9±422.3
	BioHEL-ALKR	79.7±0.3	91.5±7.1	10.3±3.6	1492.0±188.5
	C4.5	79.8±0.3	6524.9±175.1	—	—
	Naive Bayes	78.6±0.3	—	—	—
	SVM	79.6±0.4	—	—	—
Germ	BioHEL-orig	93.0±2.4	5.8±0.6	5.3±2.7	407.2±46.6
	BioHEL-ALKR	94.5±5.0	4.4±0.6	3.6±1.9	5.6±0.4
	C4.5	82.2±6.7	5.2±0.4	—	—
	Naive Bayes	79.8±6.3	—	—	—
	SVM	83.7±5.4	—	—	—
kddcup	BioHEL-orig	99.7±0.0	31.3±3.7	8.8±5.7	7409.4±1193.9
	BioHEL-ALKR	99.8±0.0	29.2±3.3	5.6±4.3	2736.1±528.2
	C4.5	100.0±0.0	708.0±100.4	—	—
	Naive Bayes	99.6±0.0	—	—	—
	SVM	99.5±0.0	—	—	—
ParMX	BioHEL-orig	100.0±0.0	402.7±18.8	9.0±0.1	19683.7±2206.3
	BioHEL-ALKR	100.0±0.0	412.4±20.9	9.0±0.0	20089.3±2085.6
	C4.5	77.0±13.6	5740.0±3006.6		
	Naive Bayes	48.9±0.5			
	SVM	76.5±0.2			
wav	BioHEL-orig	82.1±1.4	35.8±4.0	6.7±3.1	79.5±6.4
	BioHEL-ALKR	83.0±1.2	36.0±1.9	6.4±3.0	60.2±2.6
	C4.5	75.0±2.3	290.7±9.3	—	—
	Naive Bayes	80.2±1.2	—	—	—
	SVM	86.5±1.7	—	—	—

this can be another area of improvement which can benefit from recent developments in LCS research [19].

7 Conclusions and further work

This paper studied the applicability of rule-based evolutionary learning methods for large-scale datasets. Specifically, we focused on BioHEL, a iterative rule learning approach, and its attribute list knowledge representation, designed to deal efficiently with datasets with a large number of continuous attributes.

We extended the representation so as to be applicable to datasets with mixed types of attributes, i.e., continuous and discrete. This extension uses the well-known GABIL representation for discrete attributes. We also proposed an efficient matching mechanism for these datasets. Afterwards, the new representation was evaluated across a broad set of large-scale domains, containing datasets with various characteristics in what regards to the number of instances, attributes and degree of class balance that is present. The representation was compared against the previous BioHEL representation, the NAX representation (for domains with only continuous attributes) and also against three well-known machine learning methods.

The results of this comparison showed that BioHEL and ALKR are competitive against other machine learning techniques in most of the evaluated scenarios and generate solutions much more compact and, hence, interpretable, than C4.5. The evaluation process also showed some areas where BioHEL's performance can be improved, such as applying the correct degree of generality pressure in its fitness function, or its competence in datasets with class imbalance.

In future work we would like to address these two identified issues, as well as compare and combine the ALKR representation with similar efficiency-oriented matching mechanisms [9]. It would also be interesting to study mechanism for parameters self-adjusting, in order to avoid having to fine-tune BioHEL for each domains, and improve its general applicability to large-scale datasets. All these objectives will be made easier by the development of theoretical models that explain the behavior of BioHEL and its representations. We believe that these models would not only benefit BioHEL, but they would also be important milestones for the application of evolutionary learning techniques to large-scale datasets.

Acknowledgments

We would like to thank Michael Holdsworth and George W. Bassel for providing us the Germinate dataset. We are grateful for the use of the University of Nottingham's High Performance Computer.

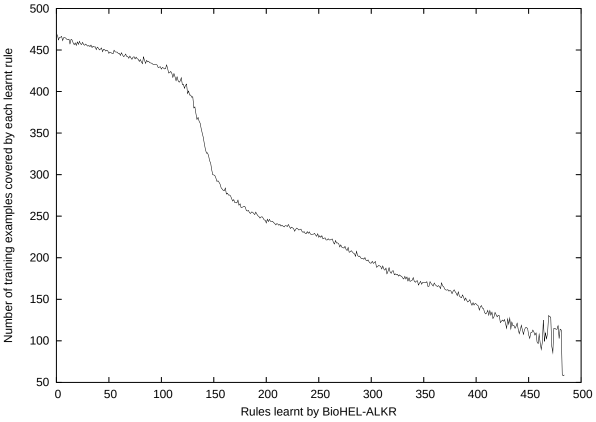
8 References

[1] J. Bacardit. *Pittsburgh Genetics-Based Machine Learning in the Data Mining era: Representations, generalization, and run-time.* PhD thesis, Ramon Llull University, Barcelona, Spain, 2004.

[2] J. Bacardit, E. K. Burke, and N. Krasnogor. Improving the scalability of rule-based evolutionary learning. *Memetic Computing*, in press, 2009.

[3] J. Bacardit and N. Krasnogor. Performance and efficiency of memetic pittsburgh learning classifier

Figure 3: Average number of training instances covered by each rule iteratively learnt by BioHEL for the ParMX dataset



systems. *Evolutionary Computation Journal*, 17(3):in press, 2009.

[4] J. Bacardit, M. Stout, J. D. Hirst, K. Sastry, X. Llorà, and N. Krasnogor. Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 346–353. ACM Press, 2007.

[5] J. Bacardit, M. Stout, J. D. Hirst, A. Valencia, R. E. Smith, and N. Krasnogor. Automated alphabet reduction for protein datasets. *BMC Bioinformatics*, 10:6, 2009.

[6] G. W. Bassel, P. Fung, T.-f. F. Chow, J. A. Foong, N. J. Provart, and S. R. Cutler. Elucidating the Germination Transcriptional Program Using Small Molecules. *Plant Physiol.*, 147(1):143–155, 2008.

[7] C. Blake, E. Keogh, and C. Merz. UCI repository of machine learning databases, 1998. (www.ics.uci.edu/mlearn/MLRepository.html).

[8] M. V. Butz. *Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design*, volume 109 of *Studies in Fuzziness and Soft Computing*. Springer, 2006.

[9] M. V. Butz, P. L. Lanzi, X. Llorà, and D. Loiacono. An analysis of matching in learning classifier systems. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1349–1356. ACM, 2008.

[10] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines.* Department of Computer Science and Information Engineering, National Taiwan University, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[11] K. A. De Jong and W. M. Spears. Learning concept classification rules using genetic algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 651–656. Morgan Kaufmann, 1991.

[12] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[13] F. Divina, M. Keijzer, and E. Marchiori. A method for handling numerical attributes in GA-based inductive concept learners. In *GECCO 2003: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 898–908. Springer-Verlag, 12-16 July 2003.

[14] A. A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* Springer-Verlag, 2002.

[15] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[16] J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. Hayes-Roth and F. Waterman, editors, *Pattern-directed Inference Systems*, pages 313–329. Academic Press, New York, 1978.

[17] X. Llorà, R. Reddy, B. Matesic, and R. Bhargava. Towards better than human capability in diagnosing prostate cancer using infrared spectroscopic imaging. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 2098–2105. ACM Press, 2007.

[18] X. Llorà and K. Sastry. Fast rule matching for learning classifier systems via vector instructions. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1513–1520. ACM Press, 2006.

[19] A. Orriols-Puig. *New Challenges in Learning Classifier Systems: Mining Rarities and Evolving Fuzzy Models.* PhD thesis, Ramon Llull University, Barcelona, Spain, 2008.

[20] J. Rissanen. Modeling by shortest data description. *Automatica*, vol. 14:465–471, 1978.

[21] C. Schumacher, M. D. Vose, and L. D. Whitley. The no free lunch and problem description length. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001*, pages 565–570. Morgan Kaufmann, 2001.

[22] M. Stout, J. Bacardit, J. D. Hirst, and N. Krasnogor. Prediction of recursive convex hull class assignments for protein residues. *Bioinformatics*, 24(7):916–923, 2008.

[23] H. Vafaie and K. A. De Jong. Genetic algorithms as a tool for feature selection in machine learning. In *Proceeding of the 4th International Conference on Tools with Artificial Intelligence*, pages 200–203, 1992.

[24] G. Venturini. SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts. In P. B. Brazdil, editor, *Machine Learning: ECML-93 - Proc. of the European Conference on Machine Learning*, pages 280–296. Springer-Verlag, 1993.

[25] S. W. Wilson. Get real! XCS with continuous-valued inputs. In L. Booker, S. Forrest, M. Mitchell, and R. L. Riolo, editors, *Festschrift in Honor of John H. Holland*, pages 111–121. Center for the Study of Complex Systems, 1999.

[26] I. H. Witten and E. Frank. *Data Mining: practical machine learning tools and techniques with java implementations.* Morgan Kaufmann, 2000.

[27] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Working Papers 95-02-010, Santa Fe Institute, Feb 1995. available at http://ideas.repec.org/p/wop/safiwp/95-02-010.html.