# BioHEL: Bioinformatics-oriented Hierarchical Evolutionary Learning

Jaume Bacardit and Natalio Krasnogor
ASAP Research Group
School of Computer Science & IT
University of Nottingham
{jqb,nxk}@cs.nott.ac.uk

October 30, 2006

## 1 Introduction

This technical report briefly describes our recent work in the iterative rule learning approach (IRL) of evolutionary learning/genetics-based machine learning. This approach was initiated by the SIA system [12] A more recent example is HIDER [1]. Our approach integrates some of the main characteristics of GAssist [4], a system belonging to the Pittsburgh approach of Evolutionary Learning, into the general framework of IRL. Our aims in developing this system are use all the good characteristics of GAssist but at the same time overcome some of the scalability limitations that it presents. The document is splitted in five parts, knowledge representation, general workflow, fitness function, some illustrative results and further work.

The system has the following characteristics:

- Each individual is a single rule, and each GA run learns one rule at a time.
- In order to learn all the rules of a domain this process is applied iteratively. At the end of each iteration the examples covered by the rule that have been learned are discarded from the training set. In this way, the GA is forced to explore other areas of the search space.
- The process is ended when no new rule can be learned. In the specific case of this system, which still uses the explicit default rule mechanism of GAssist, this means that the process is stopped when the system is unable to find a rule where the class associated to this rule is the majority class over the training set. At this point, all remaining examples are assigned to the default rule.
- The MDL-based fitness function [5], the ILAS windowing scheme [3] and the explicit default rule mechanism [6] have been inherited from GAssist.
- So far only the GABIL knowledge representation has been implemented.

## 2 Knowledge representation

### 2.1 Rule representation

Each individual is a rule, which consists of a predicate and an associated class. So far only the GABIL [10] knowledge representation has been implemented for the predicates. The associated

Figure 1: The separate-and-conquer meta learning system

**Separate-and-conquer algorithm**
**Input** : *Examples*
$Theory = \emptyset$
**While** $Examples \neq \emptyset$
    $Rule$ = FindBestRule($Examples$)
    $Covered$ = Cover($Rule$,$Examples$)
    **If** RuleStoppingCriterion($Rule$,$Theory$,$Examples$)
      **Exit while**
    **EndIf**
    $Examples = Examples \setminus Covered$
    $Theory = Theory \cup Rule$
**EndWhile**
**Output** : *Theory*

class can take as value all the classes defined in the domain but the one assigned to the default class.

### 2.2 Rule set representation

The rule set representation is a list of rules, interpreted in order as in GAssist. The initial rule set is empty and only has a default class. Unlike GAssist, we cannot use here the automatic default class determination process. Therefore, only two default class policies are allowed: majority and minority class.

## 3 General workflow

The general workflow of the system is inspied in the standard separate-and-conquer rule learning process, as represented by figure 1.

However, due to our knowledge representation, there are some differences in the model, plus some add-ons to deal with the fact that we have an stochastic rule learner. The differences are the following:

**Stop criteria** As we have a default rule, we do not have to learn rules until the training set is empty, but only until we cannot discover any more rules that are more beneficial than simply using the default class. Specifically, we stop learning when it is impossible to find a rule where the associated class is not the majority class. In the case of binary classification, this means that we stop when it is impossible to find a rule with at least 50% training accuracy.

**Multiple repetitions** As we use an stochastic algorithm to learn rules (a GA), sometimes it might happen that we do not learn the most optimal rule. Should we insert into the rule set any rule that the GA produces?. To alleviate this problem, the system will repeat the same learning process with identical training set N times. This means that we will have N candidate rules to insert, that is, the best individual of the final population of each of these N runs. This N rules are compared among them, and only the best one is inserted in the rule set.

Figure 2 contains the C++ code of the general workflow of BioHEL.

Figure 2: General workflow of BioHEL

```
instanceSet *is=new instanceSet(argv[2], TRAIN);
classifier_aggregated ruleSet;
classifierFactory cf;

do {
        classifier *best=NULL;
        for(int i=0;i<tGlobals->numRepetitionsLearning;i++) {
                classifier *bestIt=runGA();
                if(best==NULL || bestIt->compareToIndividual(best)>0) {
                        if(best) cf.deleteClassifier(best);
                        best=bestIt;
                }

                if(i<tGlobals->numRepetitionsLearning-1) {
                        is->restart();
                }
        }
        if(isMajority(*best)) {
                ruleSet.addClassifier(best);
                is->removeInstancesAndRestart(best);
        } else {
                cf.deleteClassifier(best);
                break;
        }
} while(1);
```

# 4 Fitness function

The fitness function for this kind of systems is more complicated than in GAssist, as each rule only solves a subset of the problem. It has to promote rules that are accurate but also that cover as much examples as possible. These two objectives are sometimes contradictory, specially in noisy real domains such as the bioinformatics datasets.

The BioHEL fitness function is inherited from GAssist MDL fitness function, defined as follows:

$$Fitness = TL \cdot W + EL \tag{1}$$

Where $TL$ stands for theory length (the complexity of the solution) and $EL$ stands for exceptions length (the accuracy of the solution). This fitness function has to be minimized.

$W$ is a weight that adjusts the relation between $TL$ and $EL$. BioHEL uses the automatic weight adjustement heuristic proposed for GAssist [5]. The parameters of this heuristic are adjusted as follows: Initial TL ratio: 0.25, weight relax factor: 0.90, max iterations without improvement: 10.

$TL$ is defined as follows:

$$TL(R) = \frac{\sum_i = 1^{NA} NumZeros(R_i)/Card_i}{NA} \tag{2}$$

Where $R$ is a rule, $NA$ is the number of attributes of the domain, $R_i$ is the predicate of rule $R$ associated to attribute $i$, NumZeros counts the number of bits set to zero for a given predicate in GABIL representation and $Card_i$ is the cardinality of attribute $i$. $TL$ always has a value between 0 and 1. It has been designed in this way in order to simplify the tuning of $W$. The number of zeros in the GABIL predicates are a measure of specificity. Therefore, promoting the minimization of zeros means promoting general and thus less complex rules.

The design of $EL$ is more complex for the fact mentioned above that we have to achieve an equilibrium between accuracy and coverage. We have to promote rules that cover as much examples as possible without sacrificing accuracy. In order to achieve this objective, we will design a coverage measure that promotes dramatically the fact of covering a certain minimum of examples, but that reduces it's effect after the coverage has surpassed this threshold. The measure is defined as follows:

$$EL(R) = 2 - acc(R) - coverage(R) \tag{3}$$

$$acc(R) = \frac{corr(R)}{matched(R)} \tag{4}$$

$$cov = \begin{cases} minCovRatio \cdot \frac{rawCov}{covBreak} & \text{If } rawCov < covBreak \\ minCovRatio + (1 - minCovRatio) \cdot \frac{rawCov - covBreak}{1 - rawCov} & \text{If } rawCov \geq covBreak \end{cases} \tag{5}$$

$$rawCov = \frac{matched(R)}{|T|} \tag{6}$$

Where $corr(R)$ is the number of examples correctly classified by $R$, $matched(R)$ is the number of examples matched by $R$, $minCovRatio$ is the weight given in the coverage formula to achieving the minimum coverage, $covBreak$ is the minimum coverage threshold and $|T|$ is the total number of training examples. For all the tests reported in the next section, $minCovRatio$ has 0.9 value, and $covBreak$ has value 0.0025.

# 5 Preliminary results

## 5.1 Synthetic problems

BioHEL was able to learn with 100% accuracy the multiplexers of 11 and 20 bits in 20 seconds and 1 hours, respectively. However, in neither case it was able to learn the optimal rule set because of the lack of a global supervision process (that GAssist has) and the specific definition of this domain where the system can induce non-optimal rules that have the same accuracy and coverage than some of the optimal ones.

The next test involved the hybrid multiplexer-parity problem [8] that so far had been impossible to learn with GAssist (it never obtained more than 70% accuracy). This problem has $2^18$ examples and 512 optimal rules, so the search space is quite big. With the current tests, with some more optimal settings pending to evaluate, BioHEL is able to solve this problem with 99.9% accuracy (only one wrong rule), although the obtained rule sets have around 330 rules instead of the optimal set of 256 rules plus default class. Even learning rule by rule this domain is quite complex.

## 5.2 Coordination Number datasets

This section presents some results on real bioinformatics domains, specifically of protein structure prediction. The specific domain used is called coordination number prediction [11]. Several datasets were generated from this domain. In this document we report some results on two of them. The specific definition of each dataset and the performance measures are reported in [7]. BioHEL results are compared to GAssist and also to the LIBSVM implementation of a Support Vector Machine [9].

### 5.2.1 CN1 - UL - 2 classes

BioHEL outperforms both GAssist and SVM, although the number of rules that BioHEL generates is much higher.

| — | BioHEL | GAssist | SVM |
|---|---|---|---|
| Residue-wise acc. | 73.86±0.65 | 71.93±0.60 | 73.78±0.59 |
| Protein-wise acc. | 77.44±0.83 | 75.88±0.82 | 77.40±0.82 |
| #rules | 68.43±5.46 | 2.00±0.00 | — |

### 5.2.2 CN2 - UL - 2 classes

BioHEL is performing better than GAssist but still marginally worse than SVM.

| — | BioHEL | GAssist | SVM |
|---|---|---|---|
| Residue-wise acc. | 76.91±0.53 | 76.04±0.55 | 77.30±0.52 |
| Protein-wise acc. | 79.63±0.60 | 78.96±0.70 | 79.93±0.64 |
| #rules | 86.24±4.48 | 5.0±0.1 | — |

# 6 Conclusions and further work

These first experiments show that BioHEL can be a competent evolutionary learning system, but that still there is a lot of work to do on it. Some further work lines follow:

1. Representation for real-valued attributes.

2. Other more restrictive stop criteria

3. Experiments with small datasets in order to determine a robust set of parameters

### 6.1 Representation for real-valued attributes

So far only the GABIL representation has been reimplemented from GAssist to BioHEL. The next step is to implement a real-valued representation, specially to be able to process with more sophisticated bioinformatics domains including Position-Specific Scoring Matrixes [2].

### 6.2 Other more restrictive stop criteria

When do we stop learning rules? So far the criteron is the most simple one: we stop when the new rules cannot contribute any more accuracy increase over the training set. This criterion is probably too simple and lets some over-specific rules get through, so maybe we can try to design more strict methods about deciding wether we insert a new rule in the rule set or not.

### 6.3 Experiments with small datasets in order to determine a robust set of parameters

The motivation for creating BioHEL has been to be able to solve big problems that GAssist is not able to solve. However, how would it perform in standard machine learning benchmarks? Moreover, the fitness function has a critical parameter, the coverage breakpoint. Can we find a robust set of parameters that gives competent performance across a large range of datasets?.

## References


[1] J.S. Aguilar-Ruiz, J.C. Riquelme, and M. Toro. Evolutionary learning of hierarchical decision rules. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 33(2):324–331, April 2003.

[2] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Scher, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–3402, 1997.

[3] J. Bacardit, D.E. Goldberg, M.V. Butz, X. Llorà, and J. M. Garrell. Speeding-up pittsburgh learning classifier systems: Modeling time and accuracy. In *Parallel Problem Solving from Nature - PPSN 2004*, pages 1021–1031. Springer-Verlag, LNCS 3242, 2004.

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

[5] Jaume Bacardit and Josep M. Garrell. Bloat control and generalization pressure using the minimum description length principle for a pittsburgh approach learning classifier system. In *Advances at the frontier of Learning Classifier Systems (Volume I)*. Springer-Verlag (in press), 2007.

[6] Jaume Bacardit, David E. Goldberg, and Martin V. Butz. Improving the performance of a pittsburgh learning classifier system using a default rule. In *Advances at the frontier of Learning Classifier Systems (Volume I)*. Springer-Verlag (in press), 2007.

[7] Jaume Bacardit, Michael Stout, Natalio Krasnogor, Jonathan D. Hirst, and Jacek Blazewicz. Coordination number prediction using learning classifier systems: performance and interpretability. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 247–254, New York, NY, USA, 2006. ACM Press.

[8] Martin V. Butz. *Rue-based Evolutionary Online Learning Systems: Learning Bounds, Classification and Prediction*. PhD thesis, University of Illinois at Urbana-Champaign, 2004.

[9] Chih-Chung Chang and Chih-Jen 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`.

[10] Kenneth A. DeJong and William 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.

[11] G. Pollastri, P. Baldi, P. Fariselli, and R. Casadio. Prediction of coordination number and relative solvent accessibility in proteins. *Proteins*, 47:142–153, 2002.

[12] 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, Berlin, Heidelberg, 1993.