Comparison of training set reduction techniques for Pittsburgh approach Genetic Classifier Systems

Jaume Bacardit and Josep Maria Garrell

Enginyeria i Arquitectura La Salle, Universitat Ramon Llull.
Psg. Bonanova 8, 08022-Barcelona, Catalonia, Spain.
{jbacardit,josepmg}@salleURL.edu

Abstract. In this paper we deal with the problem of reducing the computational cost of a Genetic Based Machine Learning (GBML) system based on the Pittsburgh Approach. In previous work we studied an incremental learning scheme that divided the training set in several strata and changed the used strata at each iteration. This scheme reduced the computational cost more than expected and even managed to improve the accuracy of the system. In this paper we compare our previous scheme with two alternative methods.

1 Introduction

The application of Genetic Algorithms (GA) [1] to classification problems is known as Genetic Based Machine Learning (GBML). One of the traditional ways of addressing it has been the Pittsburgh approach, exemplified by GABIL [2]. One of the main drawbacks of the systems based on this approach is the high computational cost.

One way to reduce this cost is by using a subset of the training set used for fitness computations. In a previous paper [3] we studied some incremental learning schemes that changed the subset of instances used for fitness computations through the iterations. The scheme performing better was called *Incremental Learning with Alternating Strata - ILAS* which managed to achieve better accuracy that the non-incremental system. In this paper we compare the *ILAS* scheme with two alternative methods to reduce the computational cost of the system:

– *Generation-Wise Sampling (GWS)*: Selecting a random sample of the training set for each iteration of the evolutionary process.
– *Accuracy-Classification Case Memory (ACCM)* [4]: A prototype selection method to reduce a priori the training set. The selected method comes from the Case-Based Reasoning field and it is a Case Base Maintenance method based on Rough Sets.

2 Framework

Our learning system is called GAssist (*Genetic Algorithms based claSSIfier sySTem*) and it is a Pittsburgh style classifier system descendant of *GABIL* [2]. The details of the system are described in [3].

Fig. 1. Incremental Learning with Alternating Strata

```
Procedure Incremental Learning with Alternating
          Strata
Input : Examples, NumStrata, NumIterations
Initialize GA
Reorder Examples in NumStrata parts of 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
```

3 Incremental Learning with Alternating Strata (ILAS)

This scheme splits the training examples into n strata and alternatively uses them in a round-robin manner through the iterations of the *GA* evolution process, changing the used stratum at each iteration. The pseudocode of this scheme can be seen in figure 1.

In our previous paper [3] we reported accuracy increases of the *ILAS* scheme over a non-incremental one for several datasets. Our hypothesis for this fact is based on how the population of the GA adapt itself to the environment that we create by changing the used training set at each iteration. The best way to adapt to this environment is by creating generalized individuals, as these individuals will have more chances of performing well with every stratum used. These better generalized individuals usually are more reduced than the ones generated by the non-incremental system and, therefore, faster to evaluate. This is the reason of having a computational cost reduction higher than expected.

4 Generation-Wise Sampling (GWS)

The next incremental scheme tested selects, for each generation of the *GA*, a sample of approximately $|T|/n$ instances (where $|T|$ is the size of the training set) of equal class distribution than the whole set. Therefore, its performance should be similar to the *ILAS* scheme with n strata. The code of the algorithm can be seen in figure 2.

5 ACCM

ACCM is a case base maintenance method that uses the Rough Sets theory to extract two measures of relevance from each training instance. These measures are the basis of a competence model based on the concepts of *coverage* (the set of cases that can be correctly classified by this instance) and *reachability* (the set of cases that can be used to correctly classify this case). Using these two concepts *ACCM* computes whether it is necessary to maintain or delete each instance. An extensive explanation of the algorithm can be found in [4].

6 Test suite

We have selected six problems from the well-known University of California at Irvine (UCI) repository [5]. The selected problems are: "Wisconsin Breast Cancer" (*bre*), "Ionosphere" (*ion*), "Pima-indians-diabetes" (*pim*), "Pen-Based

Fig. 2. Generation-Wise Sampling

```
Procedure Generation-Wise Sampling
Input : Examples, Reduction, Classes, NumIterations
Initialize GA
{Examples_1, Examples_2, ..., Examples_Classes} = Split
Examples into Classes sets with the examples belonging to
each class
Iteration = 0
While Iteration < NumIterations
    Sample = ∅
    If Iteration = NumIterations − 1 Then Sample = Examples
    Else
       ForEach i in Classes Do
           SampleSize = |Examples_i|/Reduction
           While SampleSize >= 1
               Select a random instance from Examples_i without replacement
               Add instance to Sample
               SampleSize = SampleSize − 1
           EndWhile
           If SampleSize > 0 Then
              If rand(0, 1) < SampleSize Then Add instance to Sample
              EndIf
           EndIf
       EndForEach
    EndIf
    Run one iteration of the GA with Sample
    Iteration = Iteration + 1
EndWhile
Output : Best individual (set of rules) from GA population
```

Recognition of Handwritten Digits" (*pen*), "SatImage" (*sat*) and "Thyroid Disease" (*thy*). The last three problems have a number of instances much higher that the others (ranging from 6435 to 10992 instances). These are selected because the computational cost becomes critical in problems like these.

The partition of the examples into the training and testing sets was done using *stratified ten-fold cross-validation*. Each test is repeated 15 times with different seeds. The *ILAS* and *GWS* schemes are tested with 2, 3 and 4 strata/reduction degrees for the small problems (*pim*,*bre* and *ion*) and 5, 10 and 20 for the big ones (*thy*,*sat* and *pen*). The *ACCM* method was not tested with the big problems because the train set reduction (81%) is not enough to achieve a reasonable computational cost. The non-incremental scheme (*NON*) is also included as a baseline.

7 Results

The aim of these tests is to compare the *ILAS* incremental learning scheme against *GWS* and *ACCM* in two aspects: accuracy and computational cost. Therefore, for each method and test problem we show the averaged cross-validation accuracy and speedup. The meaning of the speedup is the ratio between the non-incremental and incremental schemes run times. The implicit generalization pressure described previously is the reason that most speedups are higher than expected. Results are shown in table 1.

From a global point of view we can see that the *GWS* sampling scheme presents a very similar behavior to *ILAS*, in both accuracy and speedup. Looking at the results of the *ACCM* method we can see that it fells behind the other two methods in accuracy and, specially, in computational cost reduction. We think that the reason of the accuracy drop is that *ACCM* was designed as a case base maintenance (*CBM*) method from a *CBR* system using a nearest-neighbour classifier. The goal of *ACCM* method is to reduce noise and redundancy in the training set. A *GBML* system, however, benefits from redundancy because the rules that survive along the *GA* iterations are the ones that have best coverage, and redundancy helps the good rules survive because they become more general.

Table 1. Results of the comparative tests

Problem	scheme	accuracy	speedup
bre	NON	95.6%	—
	ILAS2	95.9%	2.72
	ILAS3	96.0%	4.63
	ILAS4	95.8%	5.70
	GWS2	95.8%	2.46
	GWS3	95.8%	4.43
	GWS4	95.9%	5.76
	ACCM	95.2%	1.30
ion	NON	89.5%	—
	ILAS2	90.2%	2.72
	ILAS3	90.6%	4.63
	ILAS4	91.0%	5.70
	GWS2	90.6%	2.46
	GWS3	90.9%	4.43
	GWS4	90.9%	5.76
	ACCM	89.9%	1.18
pim	NON	75.2%	—
	ILAS2	74.8%	2.67
	ILAS3	74.6%	4.41
	ILAS4	74.0%	5.85
	GWS2	74.5%	2.44
	GWS3	74.2%	4.23
	GWS4	73.9%	5.88
	ACCM	72.6%	1.31
pen	NON	79.9%	—
	ILAS5	79.9%	5.18
	ILAS10	79.4%	10.37
	ILAS20	78.9%	20.44
	GWS5	79.6%	4.92
	GWS10	79.3%	9.72
	GWS20	79.4%	18.75
sat	NON	79.9%	—
	ILAS5	79.9%	4.73
	ILAS10	79.4%	9.04
	ILAS20	78.9%	16.54
	GWS5	79.6%	4.62
	GWS10	79.3%	8.84
	GWS20	79.4%	15.80
thy	NON	93.6%	—
	ILAS5	93.7%	5.20
	ILAS10	93.6%	9.84
	ILAS20	93.5%	18.52
	GWS5	93.6%	4.24
	GWS10	93.5%	8.13
	GWS20	93.5%	15.23

8 Conclusions and further work

In this paper we have extended our prior work in reducing the computational cost of Pittsburgh approach Genetic-Based Machine Learning systems by means of incremental learning methods. This extension has consisted on two alternative methods: sampling (*Generation-Wise Sampling*) and a Case-Base Maintenance method based on Rough Sets. The tests indicate that the *GWS* method has very similar behavior to *ILAS*, and also show that the *ACCM* method is not well suited for a *GBML* system.

As a further work in studying the behavior of the *ILAS* and *GWS* methods we could analyze the diversity of the population. Also, we could refine the *GWS* method by using some statistical test to help choosing a better sample. Finally, it would be interesting to analyze the datasets with some complexity metrics in order to find a correlation between the maximum run time reduction before the accuracy drop and the characteristics of the problem.

Acknowledgments

The authors acknowledge the support provided under grant numbers 2001FI 00514, TIC2002-04160-C02-02, TIC 2002-04036-C05-03 and 2002SGR 00155. Finally we would like to thank Enginyeria i Arquitectura La Salle for their support to our research group.

References

1. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
2. DeJong, K.A., Spears, W.M., Gordon, D.F.: Using genetic algorithms for concept learning. Machine Learning **13** (1993) 161–188
3. Bacardit, J., Garrell, J.M.: Incremental learning for pittsburgh approach classifier systems. In: Proceedings of the "Segundo Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados.". (2003) 303–311
4. Salamó, M., Golobardes, E.: Hybrid deletion policies for case base maintenance. In: Proceedings of FLAIRS-2003. (2003) 150–154
5. Blake, C., Keogh, E., Merz, C.: Uci repository of machine learning databases (1998) (www.ics.uci.edu/mlearn/MLRepository.html).