Empirical evaluation of ensemble techniques for a Pittsburgh Learning Classifier System

Jaume Bacardit[1,2] and Natalio Krasnogor[1]

[1] Automated Scheduling, Optimization and Planning research group, School of Computer Science, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK {jqb,nxk}@cs.nott.ac.uk

[2] Multidisciplinary Centre for Integrative Biology, School of Biosciences, University of Nottingham, Sutton Bonington, LE12 5RD, UK

Abstract. Ensemble techniques have proved to be very successful in boosting the performance of several types of machine learning methods. In this paper, we illustrate its usefulness in combination with GAssist, a Pittsburgh-style Learning Classifier System. Two types of ensembles are tested. First we evaluate an ensemble for consensus prediction. In this case several rule sets learnt using GAssist with different initial random seeds are combined using a flat voting scheme in a fashion similar to bagging. The second type of ensemble is intended to deal more efficiently with ordinal classification problems. That is, problems where the classes have some intrinsic order between them and, in case of misclassification, it is preferred to predict a class that is close to the correct one within the class intrinsic order. The ensemble for consensus prediction is evaluated using 25 datasets from the UCI repository. The hierarchical ensemble is evaluated using a Bioinformatics dataset. Both methods significantly improve the performance and behaviour of GAssist in all the tested domains.

1 Introduction

Ensemble learning, a family of techniques established for more than a decade in the Machine Learning community, provides performance boost and robustness to the learning process by integrating the collective predictions of a set of models in some principled fashion [1]. This family of techniques covers many different approaches, the two most representative methods being Bagging [2] and Boosting [3].

This paper presents the empirical evaluation of two types of ensemble techniques that integrate the collective predictions of models generated using Learning Classifier Systems (LCS) methods. Specifically, we will use the GAssist [4] LCS, a system belonging to the Pittsburgh approach of LCSs, that has shown to generate very compact and accurate solutions for a variety of datasets [5, 4, 6–8].

The first of these two approaches will consist in a simple consensus voting of an ensemble of rule sets generated by running GAssist several times on the same dataset with different initial random seeds. This is conceptually similar to Bagging, but its implementation is even simpler.

The second type of ensemble is designed to solve problems of ordinal classification [9]. That is, when the classes of the problem have some intrinsic order. We do this with two goals in mind: (1) improving the performance of GAssist for these datasets and (2) when a misclassification occurs, to try to keep the errors localized by attempting to minimize the distance between the actual and predicted classes according to the intrinsic class order. Our ensemble approach takes a N classes dataset and generates N-1 hierarchically structured binary datasets from it. As an example, from a 10 classes dataset, first we would learn how to separate between the examples with a class ≤ 5 and examples with class > 5. Then, using the examples of class ≤ 5 we would learn how to separate between ≤ 2 and > 2. The same would happen with examples with class > 5. At the end of this process we would have a hierarchy of 9 binary classifiers. GAssist aim is to learn these N-1 binary datasets. Afterwards, the ensemble will integrate the predictions of the N-1 binary models into a final prediction of the N classes domain. This kind of ensemble has been shown to be useful for a series of Bioinformatics datasets [6, 7, 10, 11, 8, 12] which give raise to ordinal classification problems.

The rest of the paper is structured as follows: First, section 2 describes some related work and compares it to the ensemble mechanisms studied in this paper. Next, section 3 contains the main characteristics of GAssist, the Pittsburgh LCS used in this paper. Section 4 describes and evaluates the first type of ensemble studied in this paper, the consensus voting ensemble, and section 5 the second type, the hierarchical ensemble for ordinal classification. Finally, section 6 discusses our findings and suggests possible directions for future research.

will describe the conclusions and further work of the paper.

2 Related work

Usually, there are two questions that have to be addressed when building and using an ensemble that integrates the predictions of several models:

- What data is used to train each model?
- How are the individual model predictions integrated to produce a final ensemble prediction?

In the case of Bagging [2], N views of the training set are generated by a sampling with replacement procedure, and each of the N models in the ensemble is trained with one of these views. After that, the ensemble prediction process follows a simple majority voting: the ensemble will predict the most frequent class from the ones predicted by the N members of the ensemble. The first of the two types of ensembles studied in this paper shares the same decision mechanism as Bagging, a consensus voting of the generated models. The difference with Bagging lays in the way that the models are generated. In our ensemble we have a single dataset, and the different models are generated by learning this dataset feeding the GAssist LCS with different random seeds, which results in a simpler implementation.

The aim of bagging is to generate models that complement each other. This is achieved implicitly by the sampling process used to generate the models. On the other hand, Boosting [3] achieves the same aim in an explicit way. This method generates the models and the dataset in an iterative way. The first model uses the original training data, and the later models focus on learning the examples that were mis-classified by the previous models. This is achieved by weighting the instances based on their mis-classification rate on previous models. Finally, the ensemble prediction is a weighted voting process, where the weight of a model is based on its error over the training data used to generate it.

There are few examples of the use of ensembles in the LCS community. Llorà et al. [13] studied several policies to select the representative candidates from the final population of a Pittsburgh LCS, but did not integrate individuals from several populations. Also, Bull et al. [14] investigated the use of an ensemble of complete LCS populations using an island model. Their ensemble took place during the learning process, not afterwards unlike the approach of this paper.

There are several methods reported in the literature to perform ordinal classification by means of an ensemble. For instance, the method proposed by Frank and Hall [9] takes advantage of learning techniques that can produce a probability of an instance belonging to a certain class and divides the learning process of an N class ordinal domain into N-1 binary domains in the following way:

1. This method needs models that can produce class probability estimates
2. For a given domain D with ordered classes ranging from 1 to k
3. $k-1$ binary domains D_i .. D_{k-1} are generated, where the class definition for domain i will be defined by the predicate $D > i$. That is, the subdomain D_1 will predict if the class of the examples is greater than 1, D_2 will predict if the class of the examples is greater than 2, ...
4. Models for these $k-1$ domains are generated
5. For each new unseen instances, the probability that this instance belongs to each of the k classes is computed as follows:
   - $P(D = 1) = 1 - P(D > 1)$
   - $P(D = i) = P(D > i - 1) - P(D > i), 1 < i < k$
   - $P(D = k) = P(D > k - 1)$
6. The ensemble predicts the class with higher probability

This method generates $k-1$ datasets as in our hierarchical ensemble method. However all the binary datasets of this method have the same number of instances as the original dataset, while in our method some of the datasets only need to learn a subpart of the domain (a certain sub-range of the k ordinal classes) and thus only need to contain the instances of the relevant classes.

An alternative way of doing ordinal classification was proposed by Kramer et al. [15]. Instead of dividing the problem in several sub-problems and combining the models learned from them, they treat the dataset as a regression problem, using the S-CART [16] regression trees induction method, and then map the continuous predictions provided by S-CART into some of the discrete ordinal classes. Two types of policies are studied. The first of them is a simple rounding of the outputs of the unmodified S-CART into the nearest class. The second policy is to modify internally S-CART so that it produces integer predictions corresponding to the discrete ordinal classes.

3 The GAssist Learning Classifier System

GAssist [4] is a Pittsburgh Genetic–Based Machine Learning system descendant of GABIL [17]. The system applies a near-standard generational GA that evolves individuals that represent complete problem solutions. An individual consists of an ordered, variable–length rule set.

Using the rules of an individual as an ordered set to perform the match process allows the creation of very compact rule sets by the use of default rules. We use an existing mechanism [18] to explicitly exploit this issue and determine automatically the class for the default rule.

We have used the GABIL [17] rule-based knowledge representation for nominal attributes and the adaptive discretization intervals (ADI) rule representation [4] for real-valued ones. To initialize each rule, the system chooses a training example and creates a rule that guarantees to cover this example [19].

A fitness function based on the Minimum Description Length (MDL) principle [20] is used. The MDL principle is a metric applied to a theory (a rule set here) which balances its complexity and accuracy. Our specific MDL formulation promotes rule sets with as few rules as possible as well as rules containing predicates as simple as possible. The details and rationale of this fitness formula are explained in [4].

The system also uses a windowing scheme called ILAS (incremental learning with alternating strata) [21] to reduce the run-time of the system. This mechanism divides the training set into several non-overlapping strata and chooses a different stratum at each GA iteration for the fitness computations of the individuals. ILAS empirically showed in previous experiments not only to reduce the computational cost of GAssist but also to apply generalization pressure (complementary to the one applied by the MDL-based fitness function) that helped generating more compact and accurate solutions.

Parameters of the system are described in table 1.

4 Ensembles for consensus prediction

The evaluated ensemble technique follows these steps:

1. GAssist is run N times on the unmodified training set, each time using a different seed to initialize the pseudo-random numbers generator
2. From each of these N runs a rule set is extracted. This rule set corresponds to the best individual of the population, evaluated using the training set
3. For each instance in the test set, the N members of the ensemble produce a prediction. The majority class of these predictions is used

Table 1. GAssist configuration for the tests reported in the paper

Parameter	Value
General parameters	
Crossover probability	0.6
Selection algorithm	Tournament
Tournament size	3
Population size	400
Individual-wise mutation probability	0.6
Initial #rules per individual	20
Rule Deletion operator	
Iteration of activation	5
Minimum number of rules	#active rules + 3
MDL-based fitness function	
Iteration of activation	25
Initial theory length ration	0.075
Weight relax factor	0.9
ADI rule representation	
Split and merge probability	0.05
Initial reinitialize probability	0.02
Final reinitialize probability	0
#bins of uniform-width discretizers	4,5,6,7,8,10,15,20,25
Maximum number of intervals	5

This ensemble technique is very similar to Bagging, with just one difference: all models (rule sets) are learned using the same training data: the original training set.

Each rule set produced by GAssist is stored in text format as its phenotype representation: the actual ordered predicates in conjunctive normal form that constitute a rule set. Moreover, when these rule sets are dumped to text format, only the relevant attributes are expressed. This means that the ensemble code will not make unnecessary calculations for the match process of the irrelevant attributes. To illustrate the text format used to express the rules generated by GAssist, figure 1 contains an example of a rule set for the Wisconsin Breast Cancer domain generated by GAssist.

```
1:Att Clump_Thickness is [<9.4] and Att Cell_Size_Uniformity is [<4.6] and Att Cell_Shape_Uniformity is [<6.4] and Att Marginal_Adhesion is [<7.75] and Att Single_Epi_Cell_Size is [<5.5][>7.75] and Att Bare_Nuclei is [<5.5] and Att Normal_Nucleoli is [<4][5.5,8.5] and Att Mitoses is [<6.76][>7.12] → benign
2:Att Cell_Size_Uniformity is [<1.9] and Att Single_Epi_Cell_Size is [<7.75] and Att Normal_Nucleoli is [<4][5.5,8.5] → benign
3:Default rule → malignant
```

Fig. 1. Rule set generated by GAssist for the Wisconsin Breast Cancer dataset

4.1 Empirical evaluation

In order to evaluate the performance of the ensemble method described in this paper we have used a test suite of 25 datasets that represent a broad range of domains in respect to number of attributes, instances, type, etc. These problems were taken from the University of California at Irvine (UCI) repository [22], and their features are summarized in table 2.

Table 2. Features of the datasets used in this paper. #Inst. = Number of Instances, #Attr. = Number of attributes, #Real = Number of real-valued attributes, #Nom. = Number of nominal attributes, #Cla. = Number of classes, Dev.cla. = Deviation of class distribution

Dataset Properties						
Code	#Inst.	#Attr.	#Real	#Nom.	#Cla.	Dev.cla.
bal	625	4	4	—	3	18.03%
bpa	345	6	6	—	2	7.97%
bre	286	9	—	9	2	20.28%
cmc	1473	9	2	7	3	8.26%
col	368	22	7	15	2	13.04%
cr-a	690	15	6	9	2	5.51%
gls	214	9	9	—	6	12.69%
h-c	303	13	6	7	2	4.46%
hep	155	19	6	13	2	29.35%
h-h	294	13	6	7	2	13.95%
h-s	270	13	13	—	2	5.56%
ion	351	34	34	—	2	14.10%
irs	150	4	4	—	3	—
lab	57	16	8	8	2	14.91%
lym	148	18	3	15	4	23.47%
pim	768	8	8	—	2	15.10%
prt	339	17	—	17	21	5.48%
son	208	60	60	—	2	3.37%
thy	215	5	5	—	3	25.78%
vot	435	16	—	16	2	11.38%
wbcd	699	9	9	—	2	15.52%
wdbc	569	30	30	—	2	12.74%
wine	178	13	13	—	3	5.28%
wpbc	198	33	33	—	2	26.26%
zoo	101	16	—	16	7	11.82%

The datasets are partitioned using the standard stratified ten-fold cross-validation method. Three different sets of 10-cv folds have been used. Also, the experiments were repeated 15 times with different random seeds. This means that the GAssist results for each dataset included 450 runs, either by averaging the test accuracy of each of these 450 runs or by using the ensemble technique.

Student t-tests with a confidence interval of 95% were used to determine whether significant differences between the performance of the individual runs of GAssist and the ensemble of these same runs can be measured. The input data for the t-test will be the test accuracy obtained in each of the 30 test sets that we have (3x10-cv). The ensemble code produced one accuracy measure for each test set. The test accuracy of the individual runs of GAssist (15 repetitions for each data set) were computed by averaging these accuracies. The parameters of the system are the ones defined in [19].

Table 3 contains the results of the experiments performed to evaluate the ensemble technique studied in this paper. The table contains, for each dataset, the average accuracy of the 450 individual runs and the average accuracy of the 30 ensembles produced from the individual runs. We can observe how, for all datasets, the ensemble produces higher accuracy than the individual GAssist runs. The average accuracy increase is 2.5%. Also, the accuracy difference was significant in 10 of the 25 datasets, according to the t-tests.

Table 3. Results of the experiments to evaluate the consensus ensemble applied over GAssist runs. A • symbol in a row means that the ensemble was able to significantly outperform the GAssist individual runs according to the t-tests

Dataset	GAssist acc.	Ensemble acc.
bal	79.0±4.0	82.5±3.8•
bpa	62.4±7.8	65.7±7.7
bre	70.5±7.9	73.0±7.6
cmc	54.4±3.9	55.7±3.6
col	93.3±4.3	96.2±3.2•
cr-a	85.1±4.1	86.0±3.7
gls	66.8±9.5	71.9±8.0•
h-c1	80.4±5.9	83.0±5.2•
h-h	95.7±3.4	96.7±2.7
h-s	80.2±7.6	82.0±6.9
hep	89.8±8.0	93.6±5.5•
ion	92.0±5.2	93.1±5.1
irs	95.3±5.6	95.8±5.6
lab	98.1±5.4	100.0±0.0•
lym	80.8±11.2	84.4±9.9
pim	74.7±4.8	75.6±4.0
prt	47.5±6.7	52.7±6.8•
son	76.6±9.3	84.0±7.5•
thy	92.0±5.7	93.8±5.4
vot	97.1±3.3	97.6±2.7
wbcd	96.1±2.5	96.2±2.3
wdbc	94.3±3.1	95.2±2.7
wine	93.4±5.5	96.3±3.9•
wpbc	75.3±8.3	80.4±7.6•
zoo	92.1±8.0	94.1±6.3
ave	82.5±13.7	85.0±12.9

After showing the benefits of using this kind of ensemble to boost the performance of GAssist, we would like to perform some simple tests to illustrate the impact of the ensemble size in its performance. To this extend, we have reused the 15 rule sets that were previously integrated into a single ensemble to produce alternative ensembles of 5 and 10 rule sets. Three ensembles of 5 rule-sets each were created with non-overlapped rule sets (using rule sets 1-5 for ensemble 1, rule sets 6-10 for ensemble 2 and rule sets 11-15 for ensemble 3). Three ensembles of 10 rules-sets were created, this time with overlapped rule sets (using rule sets 1-10 for ensemble 1, 6-10 for ensemble 2 and 1-5,11-15 for ensemble 3). The performance of the 3 rule sets for each tested ensemble size were averaged.

Table 4 shows the results of these experiments comparing the performance of the ensembles of 5, 10 and 15 rule sets. This time we applied a different statistical tests, the Friedman test, because it is suited to compare multiple methods across different datasets. We have used the test as suggested in [23]. The test indicate that indeed there are significant performance differences between the three sizes of ensemble (with a probability of error of $1.1e^{-6}$). The Holm post-hoc test was used to compare a control method (the best method, the 15-rule-sets ensemble) against the other methods. With a confidence level of 95%, the Holm test indicated that the differences between the best ensemble and the other two ensembles are significant. Nevertheless, the average accuracy difference between the 5-rule-sets ensemble and the 15-rule-sets ensemble is only 0.6, and 1.9% over the average GAssist accuracy. This shows how with very few rule sets we can significantly boost the performance of GAssist.

Table 4. Comparing ensembles of different sizes (5, 10 and 15 rule-sets per ensemble) applied over rule-sets generated by GAssist

Dataset	5 rule sets	10 rule sets	15 rule sets
bal	81.6±2.9	82.3±3.0	82.5±3.8
bpa	64.4±6.7	65.5±6.6	65.7±7.7
bre	71.7±6.5	73.1±7.2	73.0±7.6
cmc	55.1±3.5	55.2±3.5	55.7±3.6
col	95.7±3.0	96.0±2.9	96.2±3.2
cr-a	85.9±3.8	85.7±3.8	86.0±3.7
gls	70.1±7.3	71.3±7.7	71.9±8.0
h-c1	82.5±5.0	83.0±4.7	83.0±5.2
h-h	96.2±2.6	96.3±2.6	96.7±2.7
h-s	81.6±6.7	82.0±6.5	82.0±6.9
hep	92.1±5.5	92.9±5.0	93.6±5.5
ion	93.1±4.6	93.0±4.9	93.1±5.1
irs	95.6±5.0	95.7±5.3	95.8±5.6
lab	99.3±2.1	99.7±1.3	100.0±0.0
lym	83.7±9.6	84.3±10.1	84.4±9.9
pim	75.5±3.5	75.8±3.5	75.6±4.0
prt	51.6±5.5	52.3±6.1	52.7±6.8
son	81.7±6.5	82.9±7.0	84.0±7.5
thy	93.4±5.0	93.3±5.4	93.8±5.4
vot	97.5±2.6	97.7±2.7	97.6±2.7
wbcd	96.3±2.3	96.2±2.3	96.2±2.3
wdbc	95.1±2.5	95.2±2.8	95.2±2.7
wine	95.9±3.4	96.3±3.3	96.3±3.9
wpbc	79.2±7.1	79.7±7.3	80.4±7.6
zoo	94.1±5.8	93.8±6.2	94.1±6.3
ave	84.4±13.1	84.8±12.9	85.0±12.9

5 Ensembles for ordinal classification

5.1 Motivation

The motivation for developing this kind of ensembles comes from our research in Bioinformatics, specially in Protein Structure Prediction (PSP). In this research area there are many features related to different properties of the complex 3D structure of proteins, some of these features are continuous like solvent accessibility [24]. Other features are defined as an integer, potentially having a high cardinality, such as contact number [6] or recursive convex hull [8]. Predicting these features can help improving the general problem of predicting the full 3D structure of a protein. If we have to predict these features using classification techniques we need to discretize them into a certain number of states. Therefore, we end up generating a problem of ordinal classification and, depending on the chosen number of states, potentially having a high number of classes.

5.2 Ensemble definition

Our ensemble-based approach at ordinal classification is divided in two parts: (1) decomposition of the original N classes dataset into several binary sub-datasets and (2) integration of the models generated for each dataset to produce a final N-classes prediction.

Hierarchical datasets generation The generation of the binary datasets works as follows:

1. We have an original dataset with N ordinal classes
2. We select a certain cut-point between the classes with the following criterion: we will select the cut-point that produces the most balanced sets in terms of number of instances at left and right of the cut-point
3. We generate a binary dataset from the N classes dataset: instances belonging to classes below the cut-point will be labelled as class 0. Instances belonging to classes over the cut-point will be labelled as class 1
4. The steps 2 and 3 will be repeated recursively for the instances to the left and to the right of the cut-point, until we arrive to the trivial case: having a binary dataset
5. Finally, GAssist is run several times on each of the N-1 datasets. The binary model at each node of the hierarchy will be a consensus prediction ensemble as defined in the previous section.

This process will effectively convert a N classes ordinal classification domain into N-1 binary classification domains organized in a hierarchical way. The structure of this hierarchical ensemble is represented in figure 2 for a 10 classes dataset. The root node separates between examples with a class < 5 and class ≥ 5. This node will be trained with the whole dataset, assigning class 0 to the examples of the original classes 0 through 4 and class 1 to examples with an original class ranging from 5 to 9. Then, we will take the examples with classes ranging from 0 to 4 (the examples that had class 0 at the root node) and we will train a new node with them. This node will learn how to separate between examples with a class < 3 and a class ≥ 3. Again, a binary dataset will be generated to do so. Examples of classes 0 to 2 will have class 0. Examples of classes 3 and 4 will have class 1. Afterwards, we will use the examples from the original classes 0 through

2 to learn how to separate between classes 0..1 and class 2. Class 2 is a leaf of the hierarchy and no model is needed for this branch. Finally, we would train another model using examples from classes 0 and 1 to learn how to distinguish them. The rest of the hierarchy of the ensemble would be generated and trained in a similar way.

Unlike the Frank and Hall approach [9], these N-1 datasets will not have the same number instances as the original training set: The first dataset, corresponding to the root of the hierarchy will have the same number of examples as the original training set. Then, the dataset corresponding to the left branch of the root node will contain the examples having a class less than the cut point of the root node. The dataset corresponding to the right branch of the root will have the examples having a class greater than the cut point of the root node, etc.

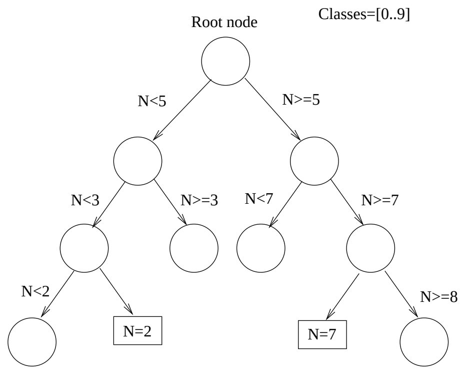

Fig. 2. Representation of the hierarchical ensemble for ordinal classification. Nodes (circles) with no descendants are already binary problems

Integration of the models into a single prediction The ensemble predictions will be performed as follows:

1. For each test instance, we will query the model at the root of the hierarchy to determine if its class is lower or higher than the first cut-point.
2. If the root model predicts class 0, the next step is to predict this instance using the model generated for the left branch of the root node
3. If the root model predicts class 1, the next step is to predict this instance using the model generated for the right branch of the root node
4. The process will continue until we reach a leaf node

This process will also be faster than the Frank and Hall method, as only $log_2(k)$ models will be queried, instead of all $k-1$ of them. Moreover, the model used at each node to predict if a given instances is lower or higher than the cut point is an ensemble itself, using the method described in the previous sections.

5.3 Empirical evaluation of the hierarchical ensemble

We will use only one domain to illustrate the performance of this hierarchical ensemble. This domain belongs to the Bioinformatics field, and specifically to protein structure prediction (PSP) [6]. Proteins are heterogeneous molecules that have a complex 3D structure very difficult to determine experimentally and therefore needs to be predicted. Several different features can be predicted from a proteins 3D structure. The domain used in this paper is called prediction of average solvent accessibility [25]. This property is real-valued and therefore we need some criterion to convert it into an ordinal set of classes. We will use the equal frequency discretization algorithm [26] for this task, dividing the domain into 10 ordinal classes.

The hierarchical ensemble obtained an accuracy of 23.9±3.0, while a flat ensemble of GAssist models performing normal classification, without any specific knowledge of the intrinsic class order, obtained an accuracy of 22.1±4.6. For reference, please note that Solvent Accessibility prediction accuracy for 10 classes with the kind of input information used in this paper ranges from 20 to 24% [25].

The performance difference is not significant, but the behaviour of both approaches is quite different: Table 5 contains the confusion matrix on one of the test folds for the hierarchical classifier, while table 6 contains the confusion matrix for the flat ensemble. The predictions of this domain are usually fed back into another PSP prediction task. Therefore it is important that, in the case of a mis-classification, the wrong predicted class is close to the real class in the intrinsic class order. The hierarchical classifier achieves this objective much better that the flat ensemble, as most of the predictions appear quite close to the diagonal. Numerically, we can compute the behaviour difference as the average misclassification penalty (AMP), defining the misclassification penalty as the distance in the intrinsic class order between the real and predicted classes of each test instance. The hierarchical ensemble obtained an AMP of 1.7±0.2, while the flat ensemble obtained an AMP of 2.0±0.2. In this case, the AMP difference between both systems was significant, according to the t-tests with 95% confidence level.

6 Conclusions and further work

This paper has empirically studied the use of ensemble techniques in combination with Learning Classifier Systems, specifically using GAssist, a Pittsburgh approach LCS. Two types of techniques are studied. The first kind of ensemble performs a Bagging-style consensus prediction, while the second one is an hierarchical ensemble intended to deal with ordinal classification domains.

Table 5. Confusion matrix of the hierarchical classifier on a test set of the average solvent accessibility domain. Each cell contains the percentage of instances of the class in the row predicted as the class in the column

		Predicted class									
		0	1	2	3	4	5	6	7	8	9
Real class	0	50.0	0.0	0.0	33.3	16.7	0.0	0.0	0.0	0.0	0.0
	1	0.0	11.1	0.0	55.6	22.2	0.0	0.0	0.0	11.1	0.0
	2	18.2	0.0	9.1	36.4	18.2	0.0	18.2	0.0	0.0	0.0
	3	0.0	10.0	10.0	30.0	30.0	0.0	0.0	10.0	10.0	0.0
	4	7.7	0.0	0.0	7.7	30.8	23.1	15.4	15.4	0.0	0.0
	5	0.0	0.0	0.0	0.0	25.0	0.0	50.0	25.0	0.0	0.0
	6	0.0	0.0	0.0	25.0	0.0	0.0	37.5	25.0	12.5	0.0
	7	0.0	0.0	0.0	0.0	0.0	11.1	22.2	22.2	44.4	0.0
	8	0.0	0.0	0.0	0.0	0.0	5.6	11.1	44.4	27.8	11.1
	9	0.0	0.0	0.0	0.0	0.0	8.3	0.0	33.3	25.0	33.3

Table 6. Confusion matrix of the flat ensemble on a test set of the average solvent accessibility domain. Each cell contains the percentage of instances of the class in the row predicted as the class in the column

		Predicted class									
		0	1	2	3	4	5	6	7	8	9
Real class	0	83.3	0.0	0.0	0.0	0.0	16.7	0.0	0.0	0.0	0.0
	1	77.8	11.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	11.1
	2	54.5	18.2	0.0	0.0	0.0	27.3	0.0	0.0	0.0	0.0
	3	50.0	30.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	20.0
	4	30.8	7.7	0.0	0.0	0.0	30.8	7.7	0.0	15.4	7.7
	5	25.0	0.0	0.0	0.0	0.0	25.0	0.0	0.0	50.0	0.0
	6	12.5	12.5	0.0	0.0	0.0	12.5	0.0	0.0	37.5	25.0
	7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	11.1	22.2	66.7
	8	0.0	0.0	0.0	0.0	0.0	11.1	0.0	0.0	27.8	61.1
	9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	16.7	83.3

These methods are not really new contributions, just variations of already existing techniques, but the experiments reported in the paper illustrate that they are very useful in combination with GAssist. The first one boots significantly the performance of GAssist on several domains, even when using very few rule sets per ensemble, while the other one helps GAssist minimize the importance of the mis-classifications, which is an issue of concern in ordinal domains.

For future work, it would be interesting to determine how can we tweak GAssist to provide the correct models to the ensemble in order to maximize the accuracy of the consensus prediction. For the hierarchical ensemble, a possible future line is to study other policies to partition the ordinal domain into several binary sub-domains, such as the ones described in the related work section. A question always open in ensemble research and very difficult to address is the interpretability capacity of the ensembles. We would like to investigate how can we improve this issue in our context. Finally, checking the accuracy-computational cost trade-off of these ensemble techniques would be very useful.

7 Acknowledgements

We acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) under grants GR/T07534/01 and EP/E017215/1. We are grateful for the use of the University of Nottingham's High Performance Computer.

References

1. various authors: Special issue on integrating multiple learned models. Machine Learning **36** (1999)
2. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
3. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning. (1996) 148–156
4. Bacardit, J.: 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. Bacardit, J., Butz, M.V.: Data mining in learning classifier systems: Comparing xcs with gassist. In: Advances at the frontier of Learning Classifier Systems. Springer-Verlag (2007) 282–290
6. Bacardit, J., Stout, M., Krasnogor, N., Hirst, J.D., Blazewicz, J.: Coordination number prediction using learning classifier systems: performance and interpretability. In: GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM Press (2006) 247–254
7. Stout, M., Bacardit, J., Hirst, J.D., Krasnogor, N., Blazewicz, J.: From hp lattice models to real proteins: Coordination number prediction using learning classifier systems. In: Applications of Evolutionary Computing, EvoWorkshops 2006, Springer LNCS 3907 (2006) 208–220
8. Stout, M., Bacardit, J., Hirst, J.D., Krasnogor, N.: Prediction of recursive convex hull class assignments for protein residues. Bioinformatics **In press** (2008)
9. Frank, E., Hall, M.: A simple approach to ordinal classification. In: Proc 12th European Conference on Machine Learning, Springer (2001) 145–156
10. Bacardit, J., Stout, M., Hirst, J.D., Sastry, K., Llora, X., Krasnogor, N.: Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO2007), London, England, ACM Press (2007) 346–353
11. Stout, M., Bacardit, J., Hirst, J.D., Blazewicz, J., Krasnogor, N.: Prediction of residue exposure and contact number for simplified hp lattice model proteins using learning classifier systems. In: Applied Artificial Intelligence, Genova, Italy, World Scientific (2006) 601–608
12. Stout, M., Bacardit, J., Hirst, J.D., Smith, R.E., Krasnogor, N.: Prediction of topological contacts in proteins using learning classifier systems. Soft Computing, Special Issue on Evolutionary and Metaheuristic-based Data Mining (EMBDM) **In Press** (2008)
13. Llorà, X., Bacardit, J., Bernadó, E., Traus, I.: Where to go once you have evolved a bunch of promising hypotheses? In: Advances at the frontier of Learning Classifier Systems (Volume I), (in press), LNAI, Springer-Verlag (2006)

14. L. Bull, M. Studley, A.J.B., Whittley, I.: On the use of rule sharing in learning classifier system ensembles. In: Proceedings of the 2005 Congress on Evolutionary Computation. (2005)
15. Kramer, S., Widmer, G., Pfahringer, B., de Groeve, M.: Prediction of ordinal classes using regression trees. Fundam. Inform. **47** (2001) 1–13
16. Kramer, S.: Structural regression trees. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), Cambridge/Menlo Park, AAAI Press/MIT Press (1996) 812–819
17. DeJong, K.A., Spears, W.M., Gordon, D.F.: Using genetic algorithms for concept learning. Machine Learning **13** (1993) 161–188
18. Bacardit, J., Goldberg, D.E., Butz, M.V.: Improving the performance of a pittsburgh learning classifier system using a default rule. In: Learning Classifier Systems, Revised Selected Papers of the International Workshop on Learning Classifier Systems 2003-2005. Springer-Verlag, LNCS 4399 (2007) 291–307
19. Bacardit, J.: Analysis of the initialization stage of a pittsburgh approach learning classifier system. In: GECCO 2005: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 2., ACM Press (2005) 1843–1850
20. Rissanen, J.: Modeling by shortest data description. Automatica **vol. 14** (1978) 465–471
21. Bacardit, J., Goldberg, D., Butz, M., Llorà, X., Garrell, J.M.: Speeding-up pittsburgh learning classifier systems: Modeling time and accuracy. In: Parallel Problem Solving from Nature - PPSN 2004, Springer-Verlag, LNCS 3242 (2004) 1021–1031
22. Blake, C., Keogh, E., Merz, C.: UCI repository of machine learning databases (1998) (www.ics.uci.edu/mlearn/MLRepository.html).
23. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7** (2006) 1–30
24. Rost, B., Sander, C.: Conservation and prediction of solvent accessibility in protein families. Proteins **20** (1994) 216–226
25. Richardson, C., Barlow, D.: The bottom line for prediction of residue solvent accessibility. Protein Eng **12** (1999) 1051–1054
26. Liu, H., Hussain, F., Tam, C.L., Dash, M.: Discretization: An enabling technique. Data Mining and Knowledge Discovery **6** (2002) 393–423