Statistical Comparisons by Means of Non-Parametric Tests: A Case Study on Genetic Based Machine Learning

Salvador García, Alberto Fernández
Dept. of Computer Science and Artificial Intelligence
Univ. of Granada
18071 Granada
salvagl@decsai.ugr.es, alberto@decsai.ugr.es

Alicia D. Benítez
Dept. of Electronic, Computer Science and Automatic Engineering
Univ. of Huelva
21071 Huelva
alicia.benitez@diesia.uhu.es

Francisco Herrera
Dept. of Computer Science and Artificial Intelligence
Univ. of Granada
18071 Granada
herrera@decsai.ugr.es

Abstract

The experimental analysis on the performance of a proposed method is a crucial and necessary task to carry out in a research. This paper is focused on the statistical analysis of the results in the community of Genetic Based Machine Learning.

Specifically, a non-parametric analysis can be performed by using the average results obtained for each data set as sample, which supposes in a simpler analysis. We use the well-known non-parametric statistical tests which can be employed and we propose the use of the most powerful statistical techniques to perform multiple comparisons among more than two algorithms.

1 Introduction

In general, the classification problem can be covered by numerous techniques and algorithms, which belong to different paradigms of Machine Learning (ML). The new developments of methods for ML must be analyzed with previous approaches by following a rigorous criterion, given that in any empirical comparison, the results depend on the choice of the cases for studying, the configuration of the experimentation and the measurements of performance. Nowadays, the statistical validation of published results is necessary to establish a certain conclusion on an experimental analysis. Statistics allows us to determine whether the results obtained are significant with respect to the choices taken and whether the conclusions remarked are supported by the experimentation carried out.

The use of statistical analysis is a necessity and we can find different studies that propose methods for conducting comparisons among various approaches [7, 12].

Evolutionary rule-based systems are a type of ML algorithms that evolve a set of rules by means of evolutionary algorithms. They receive the name of Genetic-Based Machine Learning (GBML) or Learning Classifier Systems (LCSs). We are interested in the study of the most appropriate statistical techniques for analyzing the experimentation of GBML algorithms [13, 18].

In the specialized literature, GBMLs have been analyzed by using statistical tests with the objective of computing the level of significance among the comparisons of the proposals with the other methods. The authors are usually familiarized with parametric and non-parametric tests for pairwise comparisons. Both Michigan and Pittsburgh approaches have been compared through parametric tests by means of paired t-tests by using the results obtained for all runs in each data set individually [8, 4]. The use of these type of tests is correct when we are interested

in finding the differences between two methods, but they must not be used when we are interested in comparisons that include more than two methods. In the case of repeating pairwise comparisons, there is an associated error that grows agreeing with the number of comparisons done, called the family-wise error rate (FWER), defined as the probability of at least one error in the family of hypotheses. Some authors use the Bonferroni correction for applying paired t-test in their works [14] and multiple comparisons procedures [5] for controlling the FWER in parametrical statistics.

In this paper, we show a way for doing pairwise comparisons, showing the error probability achieved when we repeat pairwise comparisons and the control of the FWER. We describe the multiple comparisons with powerful non-parametric procedures accompanied with an empirical case study which includes the comparison of 6 GBML algorithms.

In order to do that, this contribution is organized as follows. Section 2 presents the GBML algorithms used, the experimental framework and the results obtained by each method. Section 3 performs an experimental analysis based on pairwise comparisons. In the case of multiple comparisons tests, we describe and use them for analyzing the results in Section 4. Finally, Section 5 concludes the paper.

2 Genetic Based Machine Learning Algorithms for Classification

This section is divided in two parts. First we will introduce the different GBML algorithms employed in this work, giving an overall description of their characteristics, structure and operation. Finally we will present our experimental results done over 14 different data-sets from UCI repository.

2.1 GBML Methods

In this paper we use GMBL systems in order to perform classification tasks. Specifically, we have chosen 6 methods of Genetic Intervalar Rule Based Algorithms, such as Pittsburgh Genetic Intervalar Rule Learning Algorithm (Pitts-GIRLA), Supervised Inductive Algorithm (SIA), Genetic Algorithm based Classifier System (GASSIST) ADI and Intervalar, Hierarchical Decision Rules (HIDER) and XCS.

In the following we will give a brief description of the different approaches that we have employed in our work.

1. *Pittsburgh Genetic Intervalar Rule Learning Algorithm.*

   The Pitts-GIRLA Algorithm [6] is a GBML method which makes use of the Pittsburgh approach in order to perform a classification task. The main structure of this algorithm is a generational GA, in which for each generation the steps of selection, crossover, mutation and replacement are applied.

   We initialize all the chromosomes at random, with values between the range of each variable. The selection mechanism is to choose two individuals at random between all the chromosomes of the population.

   The fitness of a particular chromosome is simply the percentage of instances correctly classified by the chromosome's rule set (accuracy).

   The best chromosome of the population is always maintained as in the elitist scheme.

2. *Supervised Inductive Algorithm.*

   SIA [15] is a GBML used for obtaining of rules in classification problems.

   The conditions of the different attributes of a rule may have a "don't care" value a pair attribute-value if the attribute is symbolic or an intervalar value if the attribute is numeric.

   The main procedure of SIA begins with an empty set of rules, then it selects an example and builds the most specific rule that matches that example. By means of a GA it generalizes the condition part of the rule and deletes all the examples covered by the new rule. This procedure continues until there are no more examples to cover.

To classify a new pattern, we compute the distance of each rule to the example. Then it belongs to the class of the rule with the minimum distance measured.

3. *XCS Algorithm.*

   XCS [16] is a LCS that evolves online a set of rules that describe the feature space accurately.

   The set of rules has a fixed maximum size and it is initially built by generalizing some of the input examples, and further, they are evolved online. The result is that the knowledge is represented by a set of rules or classifiers with a certain fitness.

   When classifying unseen examples, each rule that matches the input votes according its prediction and fitness. The most voted class is chosen to be the output.

4. *GASSIST Algorithm*

   GASSIST (Genetic Algorithms based claSSIfier sySTem) [3] is a Pittsburgh-style Learning Classifier System originally inspired in GABIL from where it has taken the semantically correct crossover operator.

   The core of the system consists of a Genetic Algorithm which evolve individuals formed by a set of production rules. The individuals are evaluated according to the proportion of correct classified training examples.

   The representation for real-valued attributes is different in each approach: GASSIST-Intervalar uses intervalar rules while Adaptive Discretization Intervals Rule Representation [2] is used for GASSIST-ADI.

5. *HIDER Algorithm.*

   HIerarchical DEcision Rules (HIDER) [1], produces a hierarchical set of rules, that is, the rules are sequentially obtained and must be, therefore, tried in order until one, whose conditions are satisfied, is found.

   In order to extract the rule-list a real-coded GA is employed in the search process. Two genes will define the lower and upper bounds of the rule attribute. One rule is extracted in each iteration of the GA and all the examples covered by that rule are deleted. A parameter called Examples Pruning Factor (EPF) defines a percentage of examples that can remain uncovered. Thus, the termination criterion is reached when there are no more examples to cover, depending on EPF.

   The GA main operators are defined in the following:

   (a) *Crossover*: Where the offspring takes values between the upper and lower bounds of the parents.

   (b) *Mutation*: Where a small value is subtracted or added in the case of lower and upper bound respectively.

   (c) Fitness Function: The fitness function considers a two-objective optimization, trying to maximize the number of correctly classified examples and to minimize the number of errors.

2.2 Experimental Results

We have selected 14 data sets from UCI repository. Table 1 summarizes the properties of these data sets. It shows, for each data set, the number of examples (#Ex.), number of attributes (#Atts.) and the number of classes (#Cl.). In the case of presenting missing values (*cleveland* and *wisconsin*) we have removed the instances with any missing value before partitioning. We also add in the last columns some of the Pitts-GIRLA parameters which we have made problem-dependent in order to increase the performance of the algorithm. The rest of the parameters are the recommended by the respective authors.

The validation used is 10-fold cross validation (10fcv). We have repeated the experiments with different random seeds 5 times.

Table 2 shows the results obtained for the algorithms studied in this section over all data

Table 1: Data Sets summary descriptions and Pitts-GIRLA problem-dependent parameters

Data Set Description				Pitts-GIRLA	
Data set	**#Ex.**	**#Atts.**	**#Cl.**	#R	#Gen
bupa (bup)	345	6	2	30	5000
cleveland (cle)	297	13	5	40	5000
ecoli (eco)	336	7	8	40	5000
glass (gla)	214	9	7	20	10000
haberman (hab)	306	3	2	10	5000
iris (iri)	150	4	3	20	5000
monk-2 (mon)	432	6	2	20	5000
new-Thyroid (new)	215	5	3	20	10000
pima (pim)	768	8	2	10	5000
vehicle (veh)	846	18	4	20	10000
vowel (vow)	988	13	11	20	10000
wine (win)	178	13	3	20	10000
wisconsin (wis)	683	9	2	50	5000
yeast (yea)	1484	8	10	20	10000

sets, considering the accuracy measure in test data. The column named *Mean* shows the average accuracy achieved and the column named *SD* shows the associated standard deviation. We stress the best result for each dataset and the average one in **boldface**.

3 Performing Pairwise Comparisons

Our interest lies in presenting a way for analyzing the results offered by the algorithms in a certain study of GBML, by using non-parametric tests. This section is devoted to describe a non-parametric statistical procedure for performing pairwise comparisons between two algorithms, which is Wilcoxon's signed-rank test, Section 3.1; and to show the operation of this test in the case study presented, Section 3.2.

3.1 Wilcoxon signed-ranks test

This is the analogous of the paired t-test in non-parametric statistical procedures [13, 18]; therefore, it is a pairwise test that aims to detect significant differences between two sample means, that is, the behavior of two algorithms. It computes two sums of ranks, R^+ and R^- depending on the difference between two algorithms. If the results of the minimal of both rankings is below a certain critical value for a level of significance α, then the algorithms are significantly different. The critical value can be checked at Table B.12 in [18].

When the assumptions of the paired t-test are met, Wilcoxon's signed-ranks test is less powerful than the paired t-test. On the other hand, when the assumptions are violated, Wilcoxon's test can be even more powerful than the t-test. This allows us to apply it over the means obtained by the algorithms in each data set, without any assumptions about the sample of results obtained.

3.2 Wilcoxon's Test: A case study in GBML

In this section, we will perform the statistical analysis by means of pairwise comparisons by using the results of accuracy obtained by the algorithms described in Section 2.

In order to compare the results between two algorithms and get a determination of which one is the best, we can perform Wilcoxon signed-rank test for detecting differences in both means. This statement must be enclosed by a probability of error, that is the complement of the probability of reporting that two systems are the same, called the p-value [18]. The computation of the p-value in Wilcoxon's distribution could be carried out by a normal approximation. This test is well known and it is usually included in standard statistics packages (such as SPSS, R, etc.).

Table 3 shows the results obtained in all possible comparisons among the 6 algorithms considered in the study. We stress in bold the winner algorithm in each row when the p-value associated is below 0.05.

Table 2: Average of accuracy in test data for 10-fcv

	Pitts-GIRLA		SIA		GASSIST ADI		GASSIST Int.		HIDER		XCS	
Dat	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
bup	59.22	6.41	57.94	9.32	61.96	8.91	**65.82**	8.01	61.86	9.86	58.28	6.09
cle	**55.83**	3.76	51.94	6.84	55.18	6.38	55.42	6.00	55.45	7.23	30.18	7.24
eco	73.67	8.50	79.20	5.94	79.67	5.35	75.79	7.59	**84.22**	5.97	78.80	7.15
gla	62.47	11.04	**73.71**	10.92	63.24	10.72	62.24	10.59	69.62	13.31	57.45	10.17
hab	69.97	12.45	66.33	5.92	73.48	5.67	73.79	6.45	**74.85**	4.49	73.13	2.79
iri	94.93	5.14	95.20	4.27	96.13	3.83	94.80	3.88	**96.40**	4.09	95.07	4.22
mon	62.36	11.65	67.01	2.35	66.65	4.43	66.22	3.62	**67.19**	2.06	67.15	3.10
new	91.40	4.99	**94.51**	4.63	92.67	4.62	90.21	6.64	93.82	6.60	92.52	6.04
pim	64.85	11.61	70.13	5.82	74.60	5.39	74.44	4.48	**74.73**	4.97	74.27	5.44
veh	45.93	10.95	**69.44**	5.58	66.03	4.83	64.04	5.13	65.93	5.02	63.69	4.42
vow	24.67	5.48	**99.21**	0.74	42.48	4.70	42.44	4.90	72.48	4.82	34.04	5.43
win	70.39	21.99	**95.19**	6.05	93.02	6.06	90.90	6.88	94.76	7.92	93.94	5.82
wis	76.55	22.69	**96.69**	2.36	95.61	2.80	95.88	2.51	96.53	2.36	95.52	2.07
yea	37.23	8.77	52.24	3.67	52.93	4.15	50.19	6.27	**57.81**	3.76	36.39	5.81
Avg	63.53	10.39	**76.34**	5.31	72.40	5.56	71.58	5.92	76.12	5.89	67.89	5.41

The comparisons performed in this study are independent, so they never have to be considered in a whole. If we try to extract from Table 3 a conclusion which involves more than one comparison, we are losing control on the FWER. For instance, the statement: "The *HIDER* algorithm outperforms the *Pitts-GIRLA*, *GASSIST-ADI*, *GASSIST-Intervalar* and *XCS* algorithms with a p-value lower than 0.05" is incorrect whereas we can not prove the control of the FWER. The *HIDER* algorithm really outperforms these four algorithms considering independent comparisons.

The true statistical signification for combining pairwise comparisons is given by:

$$\begin{gathered} p = P(Reject\ H_0|H_0\ true) = \\ = 1 - P(Accept\ H_0|H_0\ true) = \\ = 1 - P(Accept\ A_k = A_i, i = 1, \ldots, k-1|H_0\ true) = \\ = 1 - \prod_{i=1}^{k-1} P(Accept\_A_k = A_i|H_0\ true) = \\ = 1 - \prod_{i=1}^{k-1} [1 - P(Reject\ A_k = A_i|H_0\ true)] = \\ = 1 - \prod_{i=1}^{k-1} (1 - p_{H_i}) \end{gathered} \tag{1}$$

From expression 1, and Table 3, we can deduce that *HIDER* is better than *Pitts-GIRLA*, *GASSIST-ADI*, *GASSIST-Intervalar* and *XCS* algorithms with a p-value of

$$p = 1 - ((1 - 0.001) \cdot (1 - 0.002) \cdot (1 - 0.008) \cdot \cdot(1 - 0.001)) = 0.012 \tag{2}$$

Hence, the previous statement has been confirmed.

On the other hand, note that the algorithm *SIA* was not included. If we include *SIA* within the multiple comparison, the error probability obtained is

$$p = 1 - ((1 - 0.001) \cdot (1 - 0.002) \cdot (1 - 0.008) \cdot \cdot(1 - 0.001) \cdot (1 - 0.331)) = 0.339 \tag{3}$$

In this case, it is not possible to declare that "*HIDER* algorithm obtains a significantly better performance than the remaining algorithms", due to the error probability achieved, $p = 0.339$, is too high.

4 Performing Multiple Comparisons

When a new proposal of GBML algorithm is developed, it could be interesting to compare it with previous proposals. Making pairwise comparisons allows us to conduct this analysis, but the experiment wise error can not be previously fixed. Moreover, a pairwise comparison is not influenced by any external factor, whereas in a multiple comparison, the set of algorithms chosen can determine the results of the analysis.

Multiple Comparisons procedures are designed for allowing us to fix the FWER before performing the analysis and for taking into account all the influences that can exist within the set of results for each algorithm. In the same way as in the previous section, the basic

Table 3: Wilcoxon's test applied over the all possible comparisons between the 6 algorithms

Comparison	R^+	R^-	p-value
Pitts-GIRLA - **GASSIST-ADI**	1	104	0.001
Pitts-GIRLA - **GASSIST-Intervalar**	10	95	0.008
Pitts-GIRLA - **HIDER**	1	104	0.001
Pitts-GIRLA - **SIA**	11	94	0.009
Pitts-GIRLA - XCS	26	79	0.096
GASSIST-ADI - GASSIST-Intervalar	80	25	0.084
GASSIST-ADI - **HIDER**	10	95	0.002
GASSIST-ADI - SIA	55	50	0.875
GASSIST-ADI - XCS	93	12	0.011
GASSIST-Intervalar - **HIDER**	10	95	0.008
GASSIST-Intervalar - SIA	38	67	0.363
GASSIST-Intervalar - XCS	73	32	0.198
HIDER - SIA	68	37	0.331
HIDER - XCS	105	0	0.001
SIA - XCS	82	23	0.064

and advanced non-parametrical tests for multiple comparisons are described in Section 4.1 and their application on the case study is conducted in Section 4.2.

4.1 Friedman test and post-hoc tests

In order to perform a multiple comparison, it is necessary to check whether all the results obtained by the algorithms present any inequality. In the case of finding it, then we can know, by using a post-hoc test, which algorithms partners average results are dissimilar. In the following, we describe the non-parametric tests used.

- The first one is Friedman's test [13], which is a non-parametric test equivalent to the repeated-measures ANOVA. Under the null-hypothesis, it states that all the algorithms are equivalent, so a rejection of this hypothesis implies the existence of differences among the performance of all the algorithms studied. After this, a post-hoc test could be used in order to find whether the control or proposed algorithm presents statistical differences with regards to the remaining methods in the comparison. The simplest of them is Bonferroni-Dunn's test, but it is a very conservative procedure and we can use more powerful tests that control the FWER and reject more hypothesis than Bonferroni-Dunn's test; for example, Holm's method [10].

  Friedman's test way of working is described as follows: It ranks the algorithms for each data set separately, the best performing algorithm getting the rank of 1, the second best rank 2, and so on. In case of ties average ranks are assigned.

  Let r_i^j be the rank of the j-th of k algorithms on the i-th of N_{ds} data sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N_{ds}} \sum_i r_i^j$. Under the null-hypothesis, which states that all the algorithms are equivalent and so their ranks R_j should be equal, the Friedman statistic:

$$\chi_F^2 = \frac{12N_{ds}}{k(k+1)} \left[\sum jR_j^2 - \frac{k(k+1)^2}{4} \right] \tag{4}$$

  is distributed according to χ_F^2 with $k-1$ degrees of freedom, when N_{ds} and k are big enough (as a rule of a thumb, $N_{ds} > 10$ and $k > 5$).

- The second one of them is Iman and Davenport's test [11], which is a non-parametric test, derived from Friedman's test, less conservative than Friedman's statistic:

$$F_F = \frac{(N_{ds}-1)\chi_F^2}{N_{ds}(K-1) - \chi_F^2} \tag{5}$$

  which is distributed according to the F-distribution with $k-1$ and $(k-1)(N_{ds}-1)$

degrees of freedom. Statistical tables for critical values can be found at [13, 18].

- Bonferroni-Dunn's test: if the null hypothesis is rejected in any of the previous tests, we can continue with Bonferroni-Dunn's procedure. It is similar to Dunnet's test for ANOVA and it is used when we want to compare a control algorithm opposite to the remainder. The quality of two algorithms is significantly different if the corresponding average of rankings is at least as great as its critical difference (CD).

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N_{ds}}}. \qquad (6)$$

The value of q_α is the critical value of Q' for a multiple non-parametric comparison with a control (Table B.16 in [18]).

- Holm's test [10]: it is a multiple comparison procedure that can work with a control algorithm (normally, the best of them is chosen) and compares it with the remaining methods. The test statistics for comparing the i-th and j-th method using this procedure is:

$$z = (R_i - R_j)/\sqrt{\frac{k(k+1)}{6N_{ds}}} \qquad (7)$$

The z value is used to find the corresponding probability from the table of normal distribution, which is then compared with an appropriate level of confidence α. In Bonferroni-Dunn comparison, this α value is always $\alpha(k-1)$, but Holm's test adjusts the value for α in order to compensate for multiple comparison and control the FWER.

Holm's test is a step-up procedure that sequentially tests the hypotheses ordered by their significance. We will denote the ordered p-values by $p_1, p_2, ...$, so that $p_1 \leq p_2 \leq ... \leq p_{k-1}$. Holm's test compares each p_i with $\alpha(k-i)$, starting from the most significant p value. If p_1 is below $\alpha(k-1)$, the corresponding hypothesis is rejected and we allow to compare p_2 with $\alpha(k-2)$. If the second hypothesis is rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all the remain hypotheses are retained as well.

- Hochberg's procedure [9]: It is a step-up procedure that works in the opposite direction to Holm's method, comparing the largest p-value with α, the next largest with $\alpha/2$ and so forth until it encounters a hypothesis that it can reject. All hypotheses with smaller p values are then rejected as well. Hochberg's method is more powerful than Holm's when the hypotheses to test are independent (in this case they are independent given that we compare a control algorithm with the remaining algorithms).

4.2 Friedman's Test: A case study in GBML

This section presents the study of applying multiple comparisons procedures to the results of the use case described above. We will use the results obtained in 10fcv and we will define the control algorithm as the best performing algorithm (which obtains the lowest value of ranking, computed through Friedman's test), *HIDER* in our case. We set the experiment wise error (level of significance) in $\alpha = 0.05$ and $\alpha = 0.10$.

First of all, we have to test whether there exist significant differences among all the means of accuracy. Table 4 shows the result of applying Friedman's and Iman-Davenport's tests. Given that the statistics of Friedman and Iman-Davenport are greater than their associated critical values, there are significant differences among the observed results with a probability error $p \leq 0.05$. Attending to these results, a post-hoc statistical analysis is needed in all cases.

Then, we will employ Bonferroni-Dunn's test to detect significant differences for the control algorithm *HIDER*. Figure 1 summarizes the ranking obtained by the Friedman

Algorithm	Ranking
Pitts-GIRLA	5.071
SIA	2.857
GASSIST-ADI	2.857
GASSIST-Intervalar	4.000
HIDER	1.714
XCS	4.500
Crit. Diff. $\alpha = 0.05$	**1.822**
Crit. Diff. $\alpha = 0.10$	**1.645**

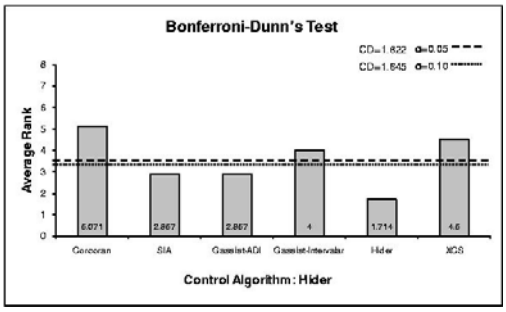

Figure 1: Rankings obtained through Friedman's test and graphical representation of Bonferroni-Dunn's procedure considering *HIDER* as control

Table 4: Results of the Friedman and Iman-Davenport Tests ($\alpha = 0.05$)

	Friedman Value	Value in χ^2	I-D Value	Value in F_F
10fcv	**30.939**	11.071	**10.297**	2.356

test and the critical difference of Bonferroni-Dunn's procedure, with the two levels of significance defined above. It also displays a graphical representation composed by bars whose height is proportional to the average ranking obtained for each algorithm. If we choose the smallest of them (which corresponds to the best algorithm), and we sum its height with the critical difference obtained by Bonferroni-Dunn (CD value), representing its result by using a cut line that goes through all the graphic, those bars above the line belong to algorithms whose behaviour are significantly worse than the contributed by the control algorithm.

We will apply more powerful procedures, such as Holm's and Hochbergs's, for comparing the control algorithm with the rest of algorithms. Table 5 shows all the possible hypotheses of comparison between the control algorithm and the remaining, ordered by their p-value and associated with their level of significance α. Both Holm's and Hochberg's procedures can be easily visualized by using the Table 5, and they coincide in the set of hypotheses rejected. Holm's method accepts the hypothesis number 2, and therefore it also accepts the number 1. Hochberg's procedure finds that the hypothesis number 3 is the first which must be rejected, so the number 4 and 5 are also rejected and the two first are maintained as accepted.

For a level of significance of $\alpha = 0.05$ and $\alpha = 0.10$, the three post-hoc tests used obtain the same result. They state that the *HIDER* algorithm outperforms *Pitts-GIRLA*, *XCS* and *GASSIST-Intervalar*. However, the power of Hochberg's or Holm's method has been pointed out as better than Bonferroni-Dunn's one. On the other hand, it is very interesting to know the exact p-value associated with each hypothesis for which it can be rejected. In the following, we will describe the method for computing these exact p-values for each test procedure, which are called "adjusted p-values" [17].

- The adjusted p-value for Bonferroni-Dunn's test (also known as the Bonferroni correction) is calculated by $p_{Bonf} = (k-1)p_i$.
- The adjusted p-value for Holm's procedure is computed by $p_{Holm} = (k-i)p_i$. Once computed all of them for all hypotheses, it is not possible to find an adjusted p-value for the hypothesis i lower than for the hypothesis j, $j < i$. In this case, the adjusted p-value for hypothesis i is set equal to the associated to the hypothesis j.
- The adjusted p-value for Hochberg's

Table 5: Holm/Hochberg Table (HIDER is the control algorithm)

i	algorithm	z	p	$\alpha/i,\ \alpha = 0.05$	$\alpha/i,\ \alpha = 0.10$
5	Pitts-GIRLA	4.748	$2.057 \cdot 10^{-6}$	0.01000	0.02000
4	XCS	3.940	$8.162 \cdot 10^{-5}$	0.01250	0.02500
3	GASSIST-Intervalar	3.232	0.00123	0.01667	0.03333
2	SIA	1.616	0.10604	0.02500	0.05000
1	GASSIST-ADI	1.616	0.10604	0.05000	0.10000

method is computed with the same formula as Holm's, and the same restriction is applied in the process, but in the opposite sense, that is, it is not possible to find an adjusted p-value for the hypothesis i lower than for the hypothesis j, $j > i$.

Table 6 shows all the adjusted p-values for each comparison that involves the control algorithm. Obviously, the procedure that needs the lowest level of confidence α for stating that *HIDER* algorithm is significantly better than the rest of methods is Hochberg's procedure, with a minimal $\alpha = 0.10604$.

In Section 3.2 we computed the p-value controlling the FWER with Wilcoxon's test, by using *HIDER* as control algorithm and under the consideration of comparing it with the remaining methods. The result obtained was $p = 0.339$.

In this case, Holm's and Hochberg's procedures are able to distinguish the *HIDER* algorithm as the best performing with a lower p-value, indicating that they are powerful tests in a multiple comparisons environment. Hochberg's method behaves the best.

5 Conclusions

In this contribution we have studied the use of statistical techniques in the analysis of the behaviour of Genetic Based Machine Learning algorithms in classification problems, analyzing the use of non-parametric statistical tests.

We have shown how to use Friedman, Iman-Davenport, Bonferroni-Dunn, Holm, Hochberg, and Wilcoxon's tests; which on the whole, are a good tool for the analysis of algorithms' performance. We have employed these procedures to carry out a comparison in a case study composed by an experimentation that involves several data sets and 6 well-known GBML algorithms.

As main conclusion on the use of non-parametric statistical methods for analyzing results, we emphasize the use of the most appropriate test depending on the circumstances and type of comparison and we recommend using two of the most powerful statistical techniques for multiple comparisons, such as Holm's and Hochberg's.

Acknowledgments

This research has been supported by the project TIN2005-08386-C05-01.

References

[1] Aguilar-Ruiz, S., Giraldez, R., Riquelme J.C., Natural Encoding for Evolutionary Supervised Learning, *IEEE Transactions on Evolutionary Computation*, In press.

[2] Bacardit, J., Garrell, J.M., Analysis and Improvements of the Adaptive Discretization Intervals Knowledge Representation, in *Genetic and Evolutionary Computation Conference (GECCO 2004)*, LNCS 3103, 2004, pp. 726-738.

[3] Bacardit, J., Garrell, J.M., 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*, LNCS 4339, 2007, pp.61-80.

[4] Bernadó-Mansilla, E., Ho., T.K., Domain of competence of XCS classifier system in complexity measurement space, *IEEE*

Table 6: Adjusted p-values by comparing *HIDER* with the remaining algorithms

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}
1	Pitts-GIRLA	$2.057 \cdot 10^{-6}$	$1.029 \cdot 10^{-5}$	$1.029 \cdot 10^{-5}$	$1.029 \cdot 10^{-5}$
2	XCS	$8.162 \cdot 10^{-5}$	$4.081 \cdot 10^{-4}$	$3.265 \cdot 10^{-4}$	$3.265 \cdot 10^{-4}$
3	GASSIST-Intervalar	0.00123	0.00615	0.00369	0.00369
4	SIA	0.10604	0.5302	0.21208	0.10604
5	GASSIST-ADI	0.10604	0.5302	0.21208	0.10604

Transactions on Evolutionary Computation, Vol. 9, No. 1, 2005, pp. 82-104.

[5] Butz, M.V., Kovacs, T., Lanzi, P.L., Wilson S.W., Toward a theory of generalization and learning in XCS, *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 1, 2004, pp. 28-46.

[6] Corcoran, A.L., Sen, S., Using real-valued genetic algorithms to evolve rule sets for classification, in *IEEE Conference on Evolutionary Computation*, 1994, pp.120-124.

[7] Demšar, J., Statistical Comparisons of Classifiers over Multiple Data Sets, *Journal of Machine Learning Research*, Vol. 7, 2006, pp. 1-30.

[8] Giráldez, R., Aguilar-Ruiz, J.S, Feature influence for evolutionary learning, in *Genetic and Evolutionary Computation Conference (GECCO 2005)*, 2005, pp.1139-1145.

[9] Hochberg, Y., A sharper Bonferroni procedure for multiple tests of significance, *Biometrika*, Vol. 75, 1998, pp. 811-818.

[10] Holm, S., A simple sequentially rejective multiple test procedure, *Scandinavian Journal of Statistics*, Vol. 6, 1979, pp. 65-70.

[11] Iman, R.L., Davenport, J.M., Approximations of the critical region of the Friedman statistic, *Communications in Statistics*, Vol. 18, 1980, pp. 571-595.

[12] Markatou, M., Tian, H., Biswas, S., Hripcsak, G., Analysis of Variance of Cross-Validation Estimators of the Generalization Error, *Journal of Machine Learning Research*, Vol. 6, 2005, pp. 1127-1168.

[13] Sheskin, D.J., *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Press. 2003.

[14] Tan, K.C., Yu, Q., Ang, J.H., A coevolutionary algorithm for rules discovery in data mining, *International Journal of Systems Science*, Vol. 37, No. 12, 2006, pp. 835-864.

[15] Venturini, G., SIA: a Supervised Inductive Algorithm with Genetic Search for Learning Attributes based Concepts, in *Machine Learning ECML-93*, LNAI 667, 1993, pp.280-296.

[16] Wilson, S.W., Classifier Fitness Based on Accuracy, *Evolutionary Computation*, Vol. 3, No. 2, 1995, pp. 149-175.

[17] Wright, S.P., Adjusted P-Values for Simultaneous Inference, *Biometrics*, Vol. 48, 1992, pp. 1005-1013.

[18] Zar, J.H., *Biostatistical Analysis*. Prentice Hall. 1999.

Mining the True Structure of Software

Fernando Berzal, Juan-Carlos Cubero, Aída Jiménez
Department of Computer Science and Artificial Intelligence
ETSIIT, University of Granada, 18071 Granada, Spain
(fberzal|jc.cubero|aidajm)@decsai.ugr.es

Abstract

When mining complex data, choosing the right representation for the underlying data is key for the practical application of data mining techniques. In the case of software systems, many program representations have already been proposed in the literature to be used by compilers and other software development tools. In this paper, we propose the use of dependence higraphs, a novel representation technique that, unlike the graph-based representation techniques commonly used by software tools, provides a hierarchical model that makes software systems suitable for the application of efficient tree mining algorithms. Moreover, our representation model is explicitly designed for making program element matching easier under a wide variety of circumstances, a task at the heart of many software mining problems.

1 Introduction

Software mining can be defined as the application of data mining tools and techniques in software engineering problems. It tries to respond to the pressing need for analyzing complex software systems. But in order to be effective, the proper underlying program representation must be used.

Many intermediate program representations are common, for instance, in compilers. In the case of compiler back ends, different intermediate representations might be used, even within the same compiler, since each representation might be better suited to perform particular kinds of transformations. Usually intended to optimize the compiled code in some sense, those transformations must preserve the semantics of the underlying program and, therefore, they must be conservative to ensure that a correct program is properly compiled.

This conservatism is due to the fact that determining the exact semantic differences between two programs is an undecidable problem. However, in software mining, the priorities are not exactly the same. Even though no automatic tool can be perfectly accurate in determining the semantic equivalence of two programs (because of the inherent undecidability of the semantic program equivalence problem), more aggressive code transformations might be useful, and even desirable, under some circumstances.

Program element matching, for example, is a common problem that must be addressed in many software mining applications. It is required for maintaining several versions of the same program (a.k.a. multi-version program analysis [17]), merging, regression testing automation, understanding the evolution of software code and the nature of software changes [24], detecting duplicated code (or near duplicates) for refactoring (or even bug fixing), and also for concept analysis [28], reverse engineering, and re-engineering.

Quite often, semantic matching is approximated by comparing the textual similarity of program elements, at their source code level. This approach resorts to string matching algorithms in order to detect the verbatim copying of arbitrary fragments of text (e.g. *diff* [13] or *bdiff* [27]) and it has been used by many 'clone