

**The Multi-label OCS with a Genetic Algorithm for
Rule Discovery: Implementation and First Results**

**Rosane M. M. Vallim
Thyago S. P. C. Duque
David E. Goldberg
Andre C. P. L. F. Carvalho
IlliGAL Report No. 2009004**

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue Urbana, IL 61801
Office: (217) 333-2346
Fax: (217) 244-5705

The Multi-label OCS with a Genetic Algorithm for Rule Discovery: Implementation and First Results

Rosane M. M. Vallim **Thyago S. P. C. Duque** **David E. Goldberg**
Andre C. P. L. F. Carvalho

Illinois Genetic Algorithms Laboratory,
University of Illinois at Urbana Champaign,
104 S. Mathews Ave, 117 Transportation Bldg, Urbana, IL
{rvallim, andre}@icmc.usp.br
{thyago, deg}@illgal.ge.uiuc.edu

Abstract

Learning Classifier Systems (LCSs) are rule-based systems that can be manipulated by a genetic algorithm. LCSs were first designed by Holland to solve classification problems and a lot of effort has been made since then, resulting in a broad number of different algorithms. One of these is called Organizational Classifier System (OCS) (Wilcox, 1995), a LCSs that tries to organize its rule set favoring good rules to be together in the same organization. However, the proposal of OCS did not include the discovery mechanism. Recently, the OCS was applied to multi-label classification (Vallim, Goldberg, Llorca, Duque, & Carvalho, 2003), a type of classification where one instance can have more than one associated label. The authors represented the multi-label classification problem as a default hierarchy and combined the organizational capabilities of OCS together with Smit's default hierarchy formation theory (Smith & Goldberg, 1992) to solve a simple multi-label problem. The purpose of this paper is to extend this idea with the inclusion of a genetic algorithm for the discovery of new rules and present some initial results obtained using the new method. The preliminary results obtained show that the method is comparable to other multi-label techniques. Final discussions present the conclusions of the work and some directions for further research.

1 Introduction

The LCSs constitute a class of learning systems that can adapt to an environment, using a knowledge base of decision rules and, oftentimes, a genetic algorithm that manipulates these rules (Butz, 2004). Although a lot of different LCSs exist in the literature, they can be divided into two main approaches: Michigan and Pittsburgh. The Michigan approach uses individuals codified as one single rule and, therefore, the final solution is formed by a set of individual rules. In this case, some niching technique has to be added to ensure the maintenance of multiple solutions in the population. In the other hand, Pittsburgh approach codifies individuals as a set of rules, with no need of niching since one individual already represents one complete solution to the problem being faced. However, the individuals are syntactically bigger than in the Michigan approach and thus the algorithm requires extra computational time and memory space.

One third approach has been proposed by Wilcox with the name of Organizational Classifier System (Wilcox, 1995). This method organizes rules in organizations based on their strength, trying to join good classifiers together. OCS did not have a genetic algorithm and as a consequence

could only organize the classifiers it already had. Vallim et al (Vallim, Goldberg, Llorca, Duque, & Carvalho, 2003) recently proposed a new LCS based on the original OCS. In (Vallim, Goldberg, Llorca, Duque, & Carvalho, 2003) the OCS capacity of joining similar rules together with Smith's theory on default hierarchy formation (Smith & Goldberg, 1992) was explored as a new way of representing and solving multi-label classification problems.

Multi-label classification is a type of classification problem where each example can be assigned an arbitrary number of labels (as opposed to one label for traditional classification). Multi-label classification is more general and oftentimes harder than single-label classification. Ambiguities in text categorization problems were the initial motivation to the research in this area. Nowadays, multi-label techniques are being used in applications as music categorization (Li & Ogihara, 2008), semantic scene classification (Boutell, Luo, Shen, & Brown, 2004), bioinformatics (Chan & Freitas, 2006) and others.

The purpose of this paper is to extend the idea proposed on (Vallim, Goldberg, Llorca, Duque, & Carvalho, 2003) with the inclusion of the discovery mechanism. We designed a genetic algorithm that works with the set of actual rules in the system and produces new rules based on it. We also propose some modifications in the original classifier system in order to make it more stable and to meet the requirements of the genetic algorithm. The proposed algorithm was tested in two different binary data sets and the results obtained were compared to other methods used for multi-label classification. These initial results show that the method proposed achieves similar performance than the other methods.

The next sections are organized as follows. Section 2 gives a brief review of the method proposed on (Vallim, Goldberg, Llorca, Duque, & Carvalho, 2003). Section 3 presents the basic aspects of genetic algorithms and how they work. Next, Section 4 proposes some improvements to the algorithm in Section 2. Section 5 presents the addition of a genetic algorithm to the improved method. The initial results obtained as well as a comparison with other multi-label methods are presented in Section 6. Finally, Section 7 brings the main conclusion of this work and the directions for further research.

2 OCS with Default Hierarchy for Multi-label Classification

In an attempt to solve multi-label classification problems, (Vallim, Goldberg, Llorca, Duque, & Carvalho, 2003) proposed a method that uses a hierarchical representation to multi-label problems together with the ideas of the OCS. The hierarchical representation consists on thinking in a solution to a multi-label classification problem as a set of rules organized as a default hierarchy, where the rules that advocate only one class are at the top of the hierarchy and act as default rules, while the rules advocating two or more classes are exception layers in the hierarchy. In a default hierarchy structure, when one exception rule matches one input all the less specific rules in the hierarchy will also match. To guarantee that the correct exception fires one have to guarantee that it has priority over the default rules. Smith's work on adaptive default hierarchy formation was used to guarantee the correct formation of the hierarchies.

The main idea of the original OCS is to separate classifiers in organizations. The mechanisms of OCS try to join classifiers with similar strengths into the same organizations with the purpose of achieving a better separation between optimal and sub-optimal classifiers. The OCS applied to multi-label classification uses this idea to try to separate not only the correct classifiers from the wrong ones, but also the correct classifiers from the partially correct ones, since in a multi-label problem one rule can assign only a portion of the set of correct labels of an instance. The final system attempts to adequately separate optimal from sub-optimal classifiers based on OCS theory

and also correctly separate the optimal rule set in a hierarchy.

The system is formed by tree main components. The first one is the production system which contains a population of organizations. Each organization contains at least 1 and at most N classifiers, where N is the total amount of classifiers in the population. Each organization has a Long-term Reputation value (LTR) and a Short-term Reputation value (STR). Classifiers carry a Reward Estimate value (RE) and a Priority Factor (PF).

The second main component is the conflict resolution and credit allocation scheme. It is responsible for first choosing which organization to select, based on the LTR values of the organizations, and then which classifier inside this organization to fire, based on the classifiers's bids. The chosen classifier sends its action to the environment and the production systems receives a reward signal that is used by the credit-allocation to update organization's and classifier's parameters.

Finally, there is the organizational growth component, responsible for sizing organizations favoring similar performing classifiers to be together. It is executed at each k iterations, and selects one of two operators (Grow and Shrink) for each one of the organizations. The Grow operator increases the size of one organization including one classifier to it, while the Shrink operator reduces the size of the organizations excluding classifiers that are performing poorly.

This system was applied to a very simple multi-label classification problem and the results showed that it was capable of putting the optimal rules together into one organization and correctly build the default hierarchy necessary to solve the problem between them.

3 Genetic Algorithms

Genetic Algorithms (GAs) (Goldberg, 1989) are search and optimization procedures based on the theory of natural selection and on basic genetic principles. They work with a population of individuals that are codified to represent solutions to a given problem. Based on the quality of these solutions the GA selects two parents, reproduce them and apply genetic operators generating new offspring, in the hope that these new solutions will be better than the actual ones. Several selection mechanisms exist and also a large number of basic genetic operators of crossover and mutation (Jong, 2006). The crossover operator exchange information between two individuals, creating a new solution that contains characteristics of both parents. Mutation randomly changes small parts of an individual performing a neighborhood search for better solutions.

In LCSs the GA is a very important component. Without the GA, a LCS can only arrange the rules it already has and qualify them.

Depending on the LCS approach the GA works in different levels (Bernadó-Mansilla, Llorá, & Traus, 2005). In Michigan LCSs the GA works selecting rules and applying basic genetic operators on them. Since what we want at the end is a set of rules that represent a solution to the classification problem, it is necessary that some niching technique is applied to guarantee the evolution and maintenance of multiple solutions in the population. In the other hand, Pittsburgh LCSs have a GA working with individuals representing a set of rules, which requires new genetic operators designed specifically to work with these individuals.

4 Improvements to the multi-label OCS

The first change to the OCS for multi-label classification was the inclusion of a validation set. The first usage for this validation set is by the Grow operator when deciding if the selected classifier will join the selected organization. Originally this decision was based on the calculation of the mean strength of all classifiers in the organization in order to verify if the classifier selected to enter this

organization had at least a percentage of the mean strength. This has been done with the purpose of not permitting a bad classifier to be inserted into a well established organization. The inclusion of the validation set has the purpose of verifying which organization is selected more frequently, if the one that the classifier is trying to enter, or the one it belongs at the moment. The number of times that each organization is selected is calculated in the validation set. The selected classifier will only change from one organization to another if the organization it belongs to has been selected less times than the other organization. This way, the system tries to avoid randomly exchanges of classifiers between two good organizations and instead tries to put some pressure towards joining all the good classifiers into only one organization. It is important to notice that this is an extra test, used in combination with the original verification based on the calculus of the mean strength.

The second usage for the validation set is in the termination criteria of the algorithm. We have noticed that because OCS is continuously trying to find new ways of organizing its rules, sometimes the system loses performance and then needs some time to recover itself. It's then important to choose an appropriate moment to stop the algorithm. We do this calculating the accuracy of the actual set of rules in the validation set. The algorithm will stop when it reaches at least a pre-defined accuracy for i consecutive times. Since we don't know *a priori* if it is really possible to reach a certain accuracy in a given problem we also have to insert another stopping criteria to prevent the system to run forever. We do this by simply stopping the system after it has seen a fixed number of input instances.

There is still another usage for the validation set, but this will be explained in Section 5.

Another change that we have made to the original system concerns the calculation of the LTR. Instead of calculating the LTR as a mean of all the rewards received by the organization until the moment, we now store the N last rewards received by the organization and use it to update the LTR each time the organization is selected to affect the environment, calculating the mean of these rewards. The purposes of this modification are related to the insertion of new organizations by the GA. The first one is that, as will be explained in Section 5, the value of N will help in the definition of the GA's time-step. The second is that we need to make sure that no matter it's an old organization or a recently created one (by means of the GA), their LTR should change at the same rate whenever it is selected to affect the environment. Restricting the update of LTR to a fixed value of N make it convenient for us to initialize the LTR of a new organization and guarantee that new and old organizations will be treated the same way. We just need to initialize the new organization's last N rewards in a way that is coherent with the value we want to initialize its LTR. Not paying attention to this could make the system to use different rates when increasing or decreasing the LTR value, because older organizations might have been selected a reasonably big amount of times.

For example, if one chooses a bad initialization of the LTR, when the GA takes place if a recently created rule is a bad one, its organization's LTR would decrease much faster than the rule's actual RE. Since the LTR falls this organization will have less and less chances to be selected and, as a consequence, the rule inside of it won't be sufficiently evaluated in order for its RE to fall down enough indicating it is actually a bad rule. Although this is a bad classifier it will seem like a reasonable one and will have chances to enter a good organization when the Grow operator takes place, which is clearly undesirable. With the new scheme this won't happen because the LTR decreases in a slower rate, giving the organization more chances to be evaluated and consequently giving enough time for the RE of the rule inside it to decrease accordingly.

We also added a covering mechanism to generate the initial population. For each example in the training set, at least one matching rule will be created, with probability P_{dc} of inserting a don't care in the antecedent. This rule will advocate the same set of labels of the example it covers. The covering mechanism can also take place during the run, in the case that no classifier in the

population matches the current input instance. In this case, one matching classifier is created advocating the same action as the input instance.

5 Inclusion of the GA

Thinking in a multi-label problem, one can have rules that have a high, medium or low strength value. The second case, medium strength, represents a rule that is receiving partial reward when it is fired, and this can happen in two different situations. The first one is simply that this rule is covering more examples than it should and has to be specialized. The second situation is when this rule actually covers exactly what it was supposed to cover but it advocates a wrong set of classes and, in this case, we have to add or subtract classes from its consequent. We can have an idea of what situation is happening by looking at the standard deviation of the rewards received by the rule. Because of the existence of these two possibilities to be explored, we designed a GA that uses two mutation operators, one in the antecedent of the rule (MA) and other in its consequent (MC).

The GA takes place at each TimeGA time-steps, selecting two parents from the actual match-set using proportionate selection with the RE as fitness. It is important to notice that the GA works with the rules that are actually in the match-set and does not consider organizations.

For each parent, the GA will calculate the standard deviation of its last M rewards and use it to calculate the probability of application of operator MA. To calculate the probability we use a simple linear interpolation between two known points. Then with this probability it will choose between the application of MA or MC. Operator MA will perform a bit-flip mutation on bit i from the parent's antecedent with probability P_m . On the other hand operator MC will add or remove a random class from the parent's consequent. One last thing has to be noticed concerning whether to choose one operator or another. Highly rewarded rules usually match only what they were supposed to match and also have the correct set of labels. What is expected is that these rules won't fall in the probability of application of MA and, therefore, MC will be selected to take place. But that isn't a very smart decision to take since there is a high probability that these rules already have the correct set of labels. So, we included one extra test before the application of MC in order to check the fitness of this rule. If it's a high rewarded rule, then the GA won't apply MC but MA. Applying MA is a better alternative in this case because if the mutation changes a specialized bit to a don't care, then the system has the opportunity of searching for more generalized rules.

After producing the new offspring, the GA will introduce them into the population. It will do this in a generational style, choosing two individuals that will be taken off the population and replacing them with the new ones. The individuals that will leave the population are the two ones with lowest fitness in the actual match-set, performing some sort of "niching replacement". Each new offspring will be inserted in the population in its own organization.

Since new organizations are created their parameters will have to be initialized. We use a positive initialization of these organizations, setting LTR to 1 with the objective of giving them chances to be selected and evaluated. Also, we initialize its N last rewards as 1, resulting in a mean of 1 which is coherent to the LTR value. As explained in Section 4 updating the LTR each time the organization is selected using the N last rewards that it has received is much more suitable when we insert new organizations in the population. The STR is also initialized as 1. The reward estimate of the new rules are initialized as 90% of their respective parent's fitness and the priority factor is initialized as 0.

The system then starts to use these new organizations and to qualify the new rules. If the new rule is a promising one, then its fitness will tend to maintain or increase. In the other hand, if it's a bad rule its fitness will start decreasing. In the case of a bad rule, the system needs enough time

to use it and to realize it is not worth it, keeping the previous solution it already had. For a GA to work accordingly one have to respect this time. Otherwise, the performance of the system will decrease. In the multi-label OCS, timing the GA is a serious problem, not only for the previous reason but also because rules keep changing organizations. A recently created rule should not go under any organizational growth component operator simply because it has not been evaluated enough yet. To find a reasonable time between GA applications (TimeGA) we divided the problem in two parts. First we calculate how many time the system needs to choose the new organization, use the new rule inside it, and realize it is a bad rule returning to its previous solution. We will call this TimeA. During TimeA the system won't be allowed to perform Grows or Shrinks. After that, we will give an extra time to the system, called TimeB, allowing it to perform these operators in the population. Just after TimeB the GA will be applied again and the process repeats.

To calculate the aproximate time after GA application that the system needs to evaluate the new rules (TimeA) we will do the following. We will calculate how many times the new organization has to be selected until it decreases to a established LTR value α . We will refer to this value as m . Considering that the new organization will be inserted in the population with a LTR value of 1 and N last rewards also equal to 1, then m is calculated as follows:

$$\frac{N - m + cm}{N} = \alpha \quad (1)$$

which can be transformed to

$$m = \left\lceil \frac{(1 - \alpha) * N}{(1 - c)} \right\rceil \quad (2)$$

where c is a constant that indicates what kind of rule we are considering. The lower the value of c , the worst the rule. Instead of counting the number of times that a rule has been selected, we decided to estimate this quantity based on conservative generality assumptions. We assume that any given rule covers at least 5% of the feature space. As a consequence, the expeted number of examples in the training set covered by a rule is at least 5% of the training set size. With this assumptions in hand, we estimate the number of repetitions n of the training set necessary to obtain realistic estimations about a rule as:

$$0.05 * N_{examples} * n = m \quad (3)$$

which can be transformed to

$$n = \left\lceil \frac{20 * m}{N_{examples}} \right\rceil \quad (4)$$

where $N_{examples}$ is the number of examples in the training set. To finish the calculation of TimeA we need to consider that with a certain percentage the system will choose a random organization from the match-set. Since we fixed this percentage in 10% then we will take this into account calculating n' as:

$$n' = \lceil 1.12n \rceil \quad (5)$$

But all the above calculation considers only one rule inserted at a time. Since our GA inserts two offspring at a time then we need double this time, because the parents are selected from the same match-set and therefore the offspring are expected to belong to the same match-set too, depending of course on the MA operator. If the two offspring belong to the same match-set then they will compete with each other everytime they match one instance, resulting in a need of being selected of $2 * n'$.

Therefore,

$$TimeA = \lceil 2 * n' \rceil \quad (6)$$

As stated before, during TimeA the system won't be allowed to perform the operators Grow and Shrink. Only after this time the operators will start to take place again for a number TimeB of repetitions of the training set.

For TimeB we consider the precision of the system on the validation set. If the system has at least a precision P on the validation set for y consecutive repetitions of the training set or if the system reaches a fixed number z of repetitions of the training set, then the GA is allowed to take place again. For example, if one chooses $y = 10$ and $z = 30$ it means that the system will apply the GA again if for 10 consecutive times the precision in the validation set remains the same, or it will apply the GA if that fact doesn't happen in 30 repetitions of the training set.

6 Experiments and Results

This section presents the experiments conducted with two different binary data sets. The first one uses an artificial multi-label data set and the second one uses a data set from the field of Bioinformatics, which was used in the work of (Chan & Freitas, 2006).

6.1 Artificial Data Set

To test the new algorithm we have designed an artificial binary multi-label data set that can be described by the following three rules:

1. 0##### \Rightarrow 1, 0
2. 1##### \Rightarrow 0, 1
3. 00##### \Rightarrow 1, 1

As can be noticed the rule set contains a default hierarchy between the first and the third rule, in accordance with the representation of multi-label classification problems as a default hierarchy presented in (Vallim, Goldberg, Llorca, Duque, & Carvalho, 2003). Table 1 presents the features of the data set originated by the above rules.

| | |
|---------------------------------|------|
| Number of examples | 52 |
| Number of attributes | 6 |
| Number of classes | 2 |
| Num. examples with label 1 | 11 |
| Num. examples with label 2 | 27 |
| Num. examples with labels 1 e 2 | 14 |
| Cardinality | 1.27 |
| Density | 0.63 |

Table 1: Features of the artificial data set used

To generate the initial population of the multi-label OCS we have used the covering mechanism with probability 0.7 of a don't care per bit of the rule's antecedent. The size of the initial population

| Method | HL | Accuracy | Precision | Recall |
|--------|--------------|--------------|--------------|--------------|
| BR | 0(0) | 1(0) | 1(0) | 1(0) |
| LP | 0(0) | 1(0) | 1(0) | 1(0) |
| ML-KNN | 0.053(0.020) | 0.947(0.020) | 0.948(0.018) | 0.990(0.015) |
| ML-OCS | 0.010(0.013) | 0.991(0.013) | 0.995(0.008) | 0.991(0.014) |

Table 2: Mean and standard deviation of 10 experiments for each metric using different methods

was 100 rules. Each rule started in a separate organization with LTR equals to 1, STR also 1, reward estimate of 0.5, and priority factor 0. N was set to 40, meaning the system stored, for each organization, its last 40 rewards. By the time of the creation of the initial organizations, the algorithm set all the N last rewards to 1. Concerning the constants used to update classifiers’ and organizations’ parameters, we used the same values used in (Vallim, Goldberg, Llorca, Duque, & Carvalho, 2003). The time between organizational growth component applications was set to 3 times the size of the training set, and the percentage of the mean strength of one organization used by the Grow and Shrink operators to 0.85. When a classifier left an organization by means of the Shrink operator, it entered a new organization with LTR 0.6. To guarantee consistency the algorithm set the N last rewards of this organization as 0.6 each. The reward estimator of the rule received a new value of 0.5 and the priority factor was set to 0. The STR of the new organization was set to the STR of the classifier’s old organization minus 0.1, to guarantee that this classifier would not have chances to reenter it’s old organizational when the Grow operator took place again.

Concerning the GA parameters, the algorithm stored the last 30 rewards of each rule for the calculation of the standard deviation, using the RE as the mean of the rewards received. We used $(x_0, y_0) = (0, 0)$, $(x_1, y_1) = (0.5, 1)$ as the two known points for interpolation, finding the probability of application of operator MA. If MA was selected it changed one bit of the individual with probability 0.3 per bit. If MC was selected, before applying it we tested the fitness of the rule, as explained in Section 5. We considered a rule highly rewarded if its fitness was above 0.9. The amount of time after GA application necessary to evaluate the new rules (TimeA) was calculated using the formula in Section 5, using $N_{examples} = 30$, $\alpha = 0.7$ and $c = 0.3$. For TimeB we used $y = 15$ and $z = 40$.

The algorithm stopped when it achieved accuracy at least 1 for 15 consecutive times in the validation set, or if the number of input instances presented to the system was greater than 350000.

We also experimented this data set with 3 other methods for multi-label classification implemented in the package Mulan¹. The methods are Binary Relevance (BR), Label Power-Set (LP) and Multi-Label KNN (ML-KNN). For the methods BR and LP we used the C4.5 algorithm as the base classifier and for the ML-KNN we used number of neighbors equals to 3. The following example-based metrics were used for evaluation (Tsoumakias & Katakis, 2007): Accuracy, Precision, Recall and Hamming Loss (HL). For each algorithm we have conducted 10 experiments using stratified 5-fold cross validation.

Table 2 presents the results of each method, where ML-OCS stands for the multi-label OCS.

The results show that, for this data set, our method achieved similar performance in all metrics when compared to the methods BR and LP. Also, it performed better than ML-KNN in all metrics. In order to verify the relevance of these results, we applied the Paired t-test (Nadeau & Bengio, 2003) to compare our method to each one of the three other methods. For all tests we used a level of confidence of 5% and the null hypothesis is that there is no estatistical diffeence between the

¹Mulan - Multi Label Classification, (<http://mlkd.csd.auth.gr/multilabel.html>)

two methods being compared. The first test compared ML-OCS and BR. The null hypothesis was accepted in all four metrics, meaning that there is no statistical difference between ML-OCS and BR in any of them. Since the metrics obtained with BR and LP were exactly the same, then the same result apply for the comparison between ML-OCS and LP. Between ML-OCS and ML-KNN the null hypothesis was rejected for the metrics HL, Accuracy and Precision. This means that there is statistical difference between ML-OCS and ML-KNN in these three metrics, and in the metric Recall there is no statistical difference between the two methods.

6.2 Bioinformatics Data Set

We conducted some experiments using a binary multi-label data set from the field of Bioinformatics. The creators of this data set designed it using cross-reference from the Uniprot (Uniprot Consortium, 2009) and Prosite (Swiss Institute of Bioinformatics, 2009) data sets. At the Uniprot data set each example represents a protein and has a reference to each Prosite pattern present in that protein. To create the data set, they used each Prosite pattern as a binary attribute, indicating the presence or absence of this pattern in the corresponding protein.

Table 3 presents the features of this data set.

| | |
|--|-------|
| Number of examples | 136 |
| Number of attributes | 153 |
| Number of classes | 2 |
| Num. examples with label 1 (Anti-oncogene) | 35 |
| Num. examples with label 2 (Apoptosis) | 95 |
| Num. examples with labels 1 e 2 | 6 |
| Cardinality | 1.044 |
| Density | 0.52 |

Table 3: Features of the data set from bioinformatics

For this data set we also conducted 10 experiments for each one of the methods used with the artificial data set, using stratified k-fold cross validation with $k = 6$. The configuration of the methods BR, LP and ML-KNN were maintained the same. For the ML-OCS we maintained almost all the parameters, with the exception of the following. The size of the initial population was set to 400 classifiers. The time between the organizational growth component application was changed to 2 times the size of the training set, once this data set is bigger than the previous one. We also changed the time between GA applications. For the calculation of TimeA we used $N_{examples} = 113$ and for TimeB we used $y = 7$ and $z = 20$. The stopping criteria also changed to reaching at least 0.7 of accuracy in the validation set for 10 consecutive times, or if this is not possible, reaching 350000 iterations.

Table 4 presents the results of the experiments using the four methods.

Applying the same statistical test with level of confidence 5% we obtained the following results. The test between BR and ML-OCS indicated acceptance of the null hypothesis in all four metrics, which means that there is no statistical difference in the results of the two methods. The same result apply for the test comparing LP and ML-OCS. The comparison between ML-KNN and ML-OCS resulted in acceptance of the null hypothesis for the metrics Accuracy, Precision and Recall. For the metric HL the test rejected the null hypothesis indicating that there is statistical difference between the two methods, with better performance of the ML-KNN.

| Method | HL | Accuracy | Precision | Recall |
|--------|--------------|--------------|--------------|--------------|
| BR | 0.290(0.004) | 0.710(0.004) | 0.732(0.004) | 0.710(0.004) |
| LP | 0.290(0.004) | 0.710(0.004) | 0.732(0.004) | 0.710(0.004) |
| ML-KNN | 0.286(0.009) | 0.711(0.015) | 0.732(0.015) | 0.712(0.015) |
| ML-OCS | 0.317(0.024) | 0.686(0.025) | 0.705(0.025) | 0.690(0.024) |

Table 4: Mean and standard deviation of 10 experiments for each metric using different methods

7 Conclusions

LCSs are adaptive systems that use a knowledge base of rules that can be manipulated by a GA. Recently, OCS and default hierarchies theory were used to construct a LCS for multi-label classification. In this work we designed a GA to work with the multi-label OCS proposed in (Vallim, Goldberg, Llorá, Duque, & Carvalho, 2003). Also, we have changed some aspects of the system in order for it to achieve better performance than before as well as to meet the requirements of the GA. The GA was designed in an attempt to understand the difficulties of multi-label classification and to explore it to produce better offspring. The resulting system was tested in two binary data sets and its performance was compared to other three multi-label methods using the Paired t-test. The results showed that there is no statistical difference among the results obtained by ML-OCS, BR and LP in all four metrics for the artificial data set. The tests with this data set also indicated the existence of statistical difference between ML-OCS and ML-KNN on three of the four metrics, with better performance of ML-OCS. For the bioinformatics data set, the test showed that there is no statistical difference between ML-OCS and the methods BR and LP in all metrics used. It also showed that between ML-OCS and ML-KNN there is no difference in three of the metrics.

This results, although preliminary, show the potential of the method proposed to solve multi-label classification problems. More tests have to be conducted using bigger and more difficult data sets to verify if the method is capable of achieving at least similar performance than other methods. One important issue that needs extra study is the scalability of the method, mostly because of the application of the organizational growth component that performs verification and applies operators to every organization in the population when it is fired. A balance between the size of the population and the time between organizational growth component application needs to be established. Another point that needs attention is the estimation of the parameters TimeA and TimeB. The calculation of TimeA is done in a very conservative way, considering always the worst case scenario. More study on the effect of this time, and also on how much conservative one should be considering ML-OCS, could bring useful insights about possible ways of reducing this time without reducing performance. In this work TimeB is set by the user, but a more detailed analysis could bring some clarifications on how much time we should give to ML-OCS so it can manipulate the new rules inserted into the system smartly.

References

- Bernadó-Mansilla, E., Llorá, X., & Traus, I. (2005). *Multiobjective Learning Classifier Systems: An Overview* (Technical Report). Urbana, IL: University of Illinois at Urbana Champaign.
- Boutell, M., Luo, J., Shen, X., & Brown, C. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*(9), 1757–1771.
- Butz, M. V. (2004). *Rule-based Evolutionary Online Learning Systems: Learning Bounds, Clas-*

- sification, and Prediction*. Doctoral dissertation, Graduate College of the University of Illinois at Urbana-Champaign, Urbana, IL.
- Chan, A., & Freitas, A. A. (2006). A New Ant Colony Algorithm for Multi-Label Classification with Applications in Bioinformatics. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 27–34).
- Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing.
- Jong, K. A. D. (2006). *Evolutionary computation: A unified approach*. MIT Press.
- Li, T., & Ogihara, M. (2008). Detecting emotion in music.
- Nadeau, C., & Bengio, Y. (2003). Inference for the generalization error. *Machine Learning*, 52(3), 239–281.
- Smith, R., & Goldberg, D. (1992). Adaptive Default Hierarchy Formation. *Applied Artificial Intelligence*, 6(1), 79–102.
- Swiss Institute of Bioinformatics (2009). Prosite - database of protein domains, families and functional sites. (Access on 01/10/2009).
- Tsoumakas, G., & Katakis, I. (2007). Multi-Label Classification: An Overview. *International Journal of Data Warehousing and Mining*, 3, 1–13.
- Uniprot Consortium (2009). Universal protein resource. (Access on 01/10/2009).
- Vallim, R., Goldberg, D., Llorca, X., Duque, T., & Carvalho, A. (2003). A new approach for multi-label classification based on default hierarchies and organizational learning. In *Proceedings of the Genetic and Evolutionary Computation Conference, Workshop Session: Learning Classifier Systems* (pp. 2017–2022).
- Wilcox, J. R. (1995). *Organizational learning within a learning classifier system*. Master's thesis, University of Illinois at Urbana-Champaign. Illigal Report No.95003.