DE GRUYTER OPEN

JAISCR, 2016, Vol. 6, No. 3, pp.173 – 188
10.1515/jaiscr-2016-0013

# SELF-CONFIGURING HYBRID EVOLUTIONARY ALGORITHM FOR FUZZY IMBALANCED CLASSIFICATION WITH ADAPTIVE INSTANCE SELECTION

Vladimir Stanovov, Eugene Semenkin, Olga Semenkina

*Institute of Computer Science and Telecommunications, Siberian State Aerospace University Krasnoyarskii rabochii ave. 31, 660014, Krasnoyarsk, Russian Federation*

**Abstract**

A novel approach for instance selection in classification problems is presented. This adaptive instance selection is designed to simultaneously decrease the amount of computation resources required and increase the classification quality achieved. The approach generates new training samples during the evolutionary process and changes the training set for the algorithm. The instance selection is guided by means of changing probabilities, so that the algorithm concentrates on problematic examples which are difficult to classify. The hybrid fuzzy classification algorithm with a self-configuration procedure is used as a problem solver. The classification quality is tested upon 9 problem data sets from the KEEL repository. A special balancing strategy is used in the instance selection approach to improve the classification quality on imbalanced datasets. The results prove the usefulness of the proposed approach as compared with other classification methods.

**Keywords**: Fuzzy classification, instance selection, genetic fuzzy system, self-configuration

## 1 Introduction

Today the area of machine learning (ML) techniques is quickly developing due to recent advances in computer and internet technologies. These advances led to the need to process, analyse and understand massive amounts of data. Modern machine learning methods often use evolutionary algorithms (EAs) as design techniques to adjust the weights or define the structure of the solution. The classical evolutionary algorithm – the genetic algorithm (GA) is a powerful method, although there is a set of specialized approaches for various problems, such as neural network structure design and weight adjustment, neuro-fuzzy inference system design, fuzzy rule base design, etc. The specialized evolutionary algorithms applied to solve machine learning problems are called genetics-based machine learning methods (GBML). A typical application of such methods is solving of classification problems [1-3].

In this paper we focus on fuzzy classification methods, i.e. fuzzy rule bases. There is a set of GBML approaches used to solve the problem of a fuzzy rule base design for classification. These methods differ in several ways: some of them optimize the positions and shape of fuzzy terms, while others optimize the rules and their combinations. The so-called Pittsburg approach, when an individual in the EA is a rule base, is used more often than the Michigan algorithm, when an individual is a single rule. However, the methods which combine both the Michigan and Pittsburg approaches seem to be the most promising.

In the case of solving real-world classification problems, researchers may face different issues related to the data available. These issues are: too large or too small amount of data available, large numbers of classes, irrelevant features and instances, errors in data measurements and missing data, imbalances in the amount of data per class and so on. These issues may cause serious difficulties in learning the actual background of the real-world process and representing it in a classification model. In this paper we focus on two main problems: a large amount of data and an imbalance in the number of instances.

The problem of a large amount of data can be solved with data reduction methods (DR). There are several groups of methods, including training set selection (TSS), active learning, instance selection (IS) and feature selection. In our work we concentrate on instance selection methods, which are used for supervised learning problems, such as classification. Instance selection and feature selection methods are divided into two groups: filter and wrapper approaches.

Instance selection is strongly connected to the problem of irrelevant instances in the training sample. Removing instances from the training set does not necessarily lead to the loss of information, especially in the case of a large number of training examples. So, the instance selection method can be not only a way to decrease the computational complexity of an algorithm, but also a method to increase the overall accuracy of the resulting model. The idea behind this study is the development of a method for selecting the instances in such a way in order to increase the learning capabilities of a GBML algorithm.

As a GBML method for our experiments, we used our modification of the hybrid fuzzy evolutionary algorithm, originally proposed by the H. Ishibuchi group. Our modifications include self-configuration, parameter tuning and some adjustments for imbalanced datasets. Unlike our previous work, in this paper we perform instance selection testing for different parameters and also consider the influence of instance selection on various classification measures. Special attention is paid to the class imbalance problem.

The rest of the paper is organized as follows: Section 2 describes the classification method, Section 3 contains the description of an instance selection approach, Section 4 contains the experimental results, and Section 5 concludes the paper.

## 2 Hybrid fuzzy GBML algorithm

The original hybrid fuzzy evolutionary algorithm was introduced by H. Ishibuchi et. al. in [4]. However, as we developed our method from scratch, we provide a short description of our implementation.

The main loop of the evolutionary algorithm implements the Pittsburg approach, i.e. each individual is a rule base. The number of rules in the rule base is not fixed and may change during the evolutionary process for every individual.

There were four different fuzzy partitions used, namely partitions into 2, 3, 4 and 5 fuzzy terms, as well as the “don’t care” condition, resulting in 15 fuzzy sets $A_0$-$A_{14}$. The fuzzy sets are presented in Figure 1.

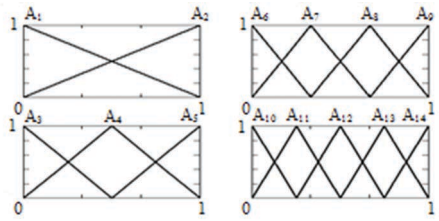

**Figure 1**. Fuzzy partitions

Each rule was represented as an integer string number from 0 to 14. The Pittsburg-type algorithm consists of the following steps:

1 Initialization using instances from the training set

2 Fitness calculation

3 Selection, Crossover, Mutation

4 Apply the Michigan-style part to each individual

5 If stopping criteria are not satisfied, go to step 2.

The Michigan part contain the following steps:

1 Define each rule in a rule base as an individual and calculate its fitness

2 Remove or add new rules to the rule base with genetic or heuristic approach

3 Return the modified rule base to the population.

Let us describe each step in detail. The initialization procedure uses instances from the training set to generate new rules. This step is important as in the case of a large number of features it is difficult to randomly generate a rule which would describe at least one instance correctly. To generate a rule, a random instance is taken from the sample, and the membership values $\mu$ are calculated for each variable for all fuzzy sets. After this, the probability of a fuzzy set to be selected is determined as

$$P(A_j) = \frac{\mu_{A_j}(x_{pi})}{\sum_{k=1}^{14} \mu_{A_j}(x_{pi})}.$$

The same procedure is repeated for all variables. Next, for every variable the fuzzy set number is changed to "Don't care" condition with 0.9 probability. After generating a rule, the confidence value is calculated using the sample available:

$$Conf(A_q \rightarrow Classk) = \frac{\sum_{x_p \in Classk} \mu_{Aq}(x_p)}{\sum_{p=1}^{m} \mu_{Aq}(x_p)}.$$

If the confidence value is larger than 0.5, then the fuzzy rule is added into the rule base.

The class number associated with a rule is not coded in the chromosome, and is determined heuristically using confidence values. For this purpose, the confidence values are calculated for all classes for every rule, and the class number corresponding to the largest confidence value is set.

The rule weight [5] is also calculated using confidence value:

$$CF_q = Conf(A_q \rightarrow Classk) - \sum_{k=1, k \neq C_q}^{M} Conf(A_q \rightarrow Classk).$$

The classification is performed by determining the winner-rule, i.e. the rule that has the largest $\mu_{Aq}(x_p)$ $CF_q$ value. The instance is classified into a class, corresponding to the winner-rule.

We generated rules 20 times for each individual, and the maximum number of rules in the algorithm was limited to 40. If all the rules received confidence values lower than 0.5, i.e. the rule base was empty, than the generation procedure was repeated.

The fitness value for all individuals was calculated as a combination of three main criteria, the training sample error $f_1(i)$, the number of rules $f_2(i)$, and the total length of all rules $f_3(i)$. The training sample error was the percentage with weight coefficient $w_1$=100, the two other criteria were used with weights $w_2$=1 and $w_3$=1.

For the selection step we applied three classical selection schemes, i.e. fitness proportional, rank and tournament selection with tournament size of 2.

There was only one specialized crossover operator used, which combines two rule bases to produce one offspring. For this purpose, the number of rules for the offspring is determined as a random number from *1* to $|S_1|+|S_2|$, where $|S_i|$ is the number of rules for individual *i*. If $|S_1|+|S_2|$exceeds the maximum number of rules (i.e. 40 in our computational experiments) then the number of rules is set to be equal to this maximum number. After this, the rule base is filled with new rules in a random way from a general rule pool created from parents' rules.

The mutation step is similar to the one used in GAs. The probability of a gen (fuzzy set) to be changed is defined as $1/(n \cdot jS_1|)$ for average mutation, $3/(n \cdot jS_1|)$ for strong mutation and $1/(3 \cdot n \cdot jS_1|)$ for weak mutation, where $n$ is the number of variables. Each of the fuzzy sets in a rule base could mutate, including "Don't care" conditions. If the confidence value of a fuzzy rule after the use of the mutation operator becomes lower than 0.5, the rule is excluded from the rule base. If there are no rules left after mutation then several rules are generated using the same procedure as at the initialization step.

The Michigan-style part is applied to every rule base after the mutation operator. At the first step, the fitness values are calculated for every rule. The rule fitness is equal to the number of instances correctly classified with this rule. If there are two identical rules, only one of them gets the non-zero fitness value. There were three types of the Michigan

part: adding new rules, deleting rules and replacing rules, i.e. first deleting, then adding. In the case of deleting rules, the number of rules $k$ to be deleted is defined as $5 \cdot (k-1) < |S| < 5 \cdot k$. The rules with the lowest fitness values are deleted first. In the case of adding new rules, the number of rules to be added is defined in the same way as for deleting, but if the number of rules exceeds the maximum number of rules then no rule is added.

New rules are added with the use of two different methods, heuristic and genetic ones. The heuristic method uses incorrectly classified instances to generate new rules using the same procedure as at the initialization step. The genetic method uses rules from the rule base to produce new rules with the tournament selection, uniform crossover and average mutation as in the genetic algorithm.

One of the modifications of the original algorithm was the self-configuration procedure which was implemented for selection, mutation, the Michigan part and adding rules in the Michigan part. The self-configuration method was first introduced in [6, 7] and was successfully applied to a similar problem in [8]. The main idea of the method is the assigning to genetic operators of the probabilities to be used in the future based on their success in the past. The method uses averaged fitness values of offspring generated by a certain operator to select a winner-operator at each generation. The winner's probability increases, while all other operators get their probabilities decreased.

We will describe this method in detail. Let $z$ be the number of different operators of the $i-th$ type. The starting probability values are set to $p_i = 1/z$. The success estimation for every type of operator is performed based on the averaged fitness values:

$$AvgFit_i = \frac{1}{2}\sum_{j=1}^{n_i} f_{ij}, i = 1,2,\ldots,z,$$

where $n_i$ is the number of offspring formed with the *i-th* operator, $f_{ij}$ is the fitness value of the *j-th* offspring, produced with the *i-th* operator, $AvgFit_i$ is the average fitness of the solutions, produced with the *i-th* operator. Then the probability of applying the operator, whose $AvgFit_i$ value is the highest among all the operators of this type, is increased by $(z \cdot K - K)/(z \cdot N)$, and the probabilities of applying other operators are decreased by $K/(z \cdot N)$, where $N$ is the number of evolutionary algorithm generations, $K$ is a constant value usually equal to 0.5.

# 3 Adaptive instance selection algorithm for imbalanced classification problems

As we mentioned in the Introduction, data reduction does not necessary lead to lower classification quality. In some cases excluding instances from the training set may lead to classification quality improvement, because the deleted instances were noisy, repeated many times and so on.

So, most of the instance selection methods are focused on both data reduction and classification quality improvement [9, 10]. However, these methods are mainly designed to create a subset only once, and not to train an accurate classifier.

The proposed instance selection algorithm is designed for learning algorithms which use a lot of iterations during the learning process. This method does not require any data preprocessing, it is not based on the $k-NN$ method and does not require the distance to be calculated between instances. Instead, instances are selected based on classification quality, i.e. it implements the wrapper approach.

Let us describe the idea in detail. At the first stage, a subsample of the training sample is created, having a fixed size (set by user). At this step, the instances are selected with equal probabilities. After this, the learning process starts for a number of generations (iterations), called the adaptation period. Only the training subset is used.

Here every instance receives a counter value $U_i$, which means the number of successful uses of the *i-th* instance. At the beginning, all $U_i$ = 1, *i=1...n*, and then are changed for every instance. At the end of the adaptation period, the best current solution, i.e. best classifier, for the subsample is used to update the counter values. Only the counters of the instances which are in the subsample are updated. If an instance j was classified correctly, then $U_j$=$U_j$+1, otherwise $U_j$=1.

So, the sample instances which were classified correctly get their counters updated, while the counters for incorrectly classified measurements are reset. After the update of counters, a new subsample

is created, using new counters. The probability for an instance $i$ to be selected is calculated using the equation:

$$p_i = \frac{1/U_i}{\sum_{j=1,n} 1/U_j} .$$

According to this equation, increasing the counter leads to a decrease in the probability of an instance being included into the subsample. The denominator in this case is needed for normalization.

Thus, the adaptive instance selection algorithm assigns lower probabilities for instances which are easier to classify. At the same time, instances which are difficult to classify or those which have not been used before get larger probabilities to be selected in the new training sample. This procedure implements two main principles: the exploration of areas of the feature space unknown before, and using information about classification quality to build a better separation between classes.

During the learning process, the best solution for every subsample is changed after every adaptation period, as it depends on the instances selected. Because of this, the probabilities are also changed, as the best solution for the subsample may present different results for the whole training set. As the sample constantly changes, different solutions can be received, improving the search process. In the case of an evolutionary algorithm being used as a learning method, during every new adaptation period, a population of solutions from the previous step is saved. Individuals in this population are able to classify instances of the new training set, but on most occasions with lower accuracy.

At each generation of the evolutionary algorithm, the best solution found for the subsample is checked on the whole sample. This step is required to exclude losing the best found solution. Moreover, the best solution for the whole training set is included into population together with the best solution for the subsample. At the end of the adaptation period, all individuals of the current population are checked on the whole training sample. This step is required as the population may contain other solutions which could have even better generalization than the best solution for the current subsample.

One more important issue during creating a subsample is setting the amount of instances to be selected for every class. If the described procedure will be used without taking the original class distribution into account, the distribution in the subsample may significantly differ. Because of this, the number of instances available for every class has to influence the subsample creation procedure. The problem of imbalanced classification with fuzzy rule bases has been previously studied in [11, 12].

One of the methods for taking this distribution into consideration is a stratified approach, which is commonly used in cross-validation. This approach is important for balanced problems, as it saves the class distribution.

However in the case of solving imbalanced classification problems, the stratified approach may be not the best way. As the described adaptive instance selection approach may choose different groups of instances, for imbalanced problems it is possible to sample more balanced subsets, than the original set. This idea is implemented in the balancing approach, which creates the subset so that the number of instances would be as balanced as possible. In this case, the minority classes may be entirely included into the training set.

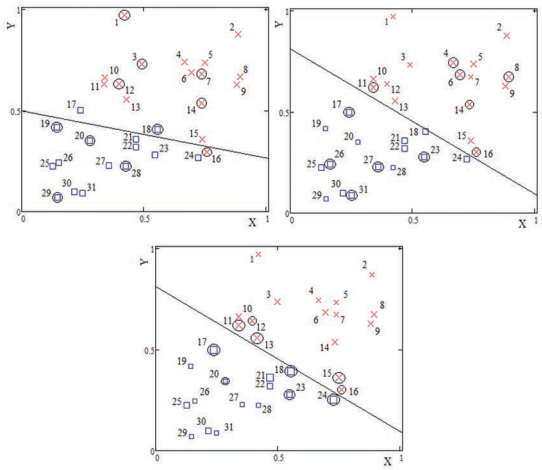

**Figure 2**. Example of instance selection

In Figure 2, we show a graphical example of which of the instances are chosen by the algorithm and how the active instance selection procedure leads to an improvement in the classification quality. The example is a two-class problem with 31 measurements, with different class instances shown with crosses and squares. On the left side, the start of the algorithm is shown, where all instances have the same probability of being selected, shown by the size of the points. The measurements in circles

are the instances that have been selected into the current subsample. The separating surface obtained by the end of the adaptation period does not classify all patterns correctly: instances 16 and 18, which are in the subsample, are misclassified. On the right side, the next adaptation period is shown. Instances 16 and 18 did not change their size (i.e. $U_{16} = U_{18} = 1$) compared to unused instances, while all the rest (1, 3, 7, 12, 14, 19, 20, 28, 29), used and correctly classified, received a lower probability and thus a smaller size.

The lower graph in Figure 2 demonstrates the situation after several adaptation periods, where those instances that are close to the separating plane between classes have higher probabilities of being chosen (and they are actually chosen at this iteration), while the remaining instances that lie further away have lower probablilties.

## 4 Experimental setup and results

We performed a set of computational experiments with the presented adaptive instance selection algorithm and hybrid fuzzy GMBL algorithm to evaluate their effectiveness. The experiments were performed on a 4-core Intel Core-i7 2600K@4400MHz processor, the program system was implemented in C++ with GCC 4.8.1 compiler, only standard C++ libraries used.

The parameters of the hybrid evolutionary fuzzy classification algorithm were set in the same way for all experiments: the population size was equal to 100, the number of generations to 10000 and the maximum number of rules to 40. The parameters for the instance selection, i.e. the size of the subsample and the length of the adaptation period varied. The subsample size was set to 5%, 10%, 15%, 20%, 25% and 30% of the original training set, the length of the adaptation period was set to 50, 100, 200 and 400 generations. Also two sampling strategies were tested – a stratified and a balancing strategy for the subsample. For each combination of parameters a 10-fold cross-validation procedure was performed twice, so that the classification quality measures were averaged over 20 runs. The standard algorithm without instance selection has also been tested.

The classification problems for testing were taken from the UCI [13] and KEEL [14] repositories. The selected problems have a large number of instances, variables and classes, some of which are highly imbalanced. The parameters of the datasets are presented in Table 1.

**Table 1**. Datasets used

| Dataset | Number of instances | Number of features | Number of classes |
|---|---|---|---|
| **Magic** | **19020** | **10** | **2** |
| **Page-blocks** | **5472** | **10** | **5** |
| **Penbased** | **10992** | **16** | **10** |
| **Phoneme** | **5404** | **5** | **2** |
| **Ring** | **7400** | **20** | **2** |
| **Satimage** | **6435** | **36** | **6** |
| **Segment** | **2310** | **19** | **7** |
| **Texture** | **5500** | **40** | **11** |
| **Twonorm** | **7400** | **20** | **2** |

The error values in Table 2 are the error rates in percentages. The next table contains the best results with instance selection and the stratified strategy.

The best instance selection configurations for the stratified strategy were the following, for Magic: 30% of the training sample and adaptation period length of 400 generations; for Page-blocks: 30%, 200; for Penbased: 15%, 50; Phoneme: 25%, 200; Ring: 25%, 50; Satimage: 30%, 50; Segment: 30%, 50; Texture: 25%, 50; Twonorm: 30%, 100. So, for most of the problems the best results were obtained with the maximum subsample size.

The difference between the standard method and instance selection with the stratified strategy is presented in Table 4.

The time ratio was calculated as a ratio of the modified algorithm to the standard algorithm in percentages. The maximum difference in accuracy was 3.59% (Penbased problem). Only for the Phoneme problem was the effect of instance selection negative (i.e. 1.26% lower for the training sample). The time spent by the modified algorithm was from 18% to 32% of the original, depending mainly on the size of the subsample. Results for the balancing strategy are presented in Table 5.

Table 6 contains the comparison of Tables 2 and 5.

**Table 2**. Results for the standard algorithm

| Dataset | Training error | Test error | Number of rules | Rule length | Time(minutes) |
|---|---|---|---|---|---|
| **Magic** | 15.06 | 15.73 | 12.6 | 3.82 | 370.62 |
| **Page-blocks** | 3.52 | 3.96 | 10.1 | 3.49 | 94.48 |
| **Penbased** | 7.06 | 7.46 | 30.5 | 6.23 | 385.2 |
| **Phoneme** | 15.03 | 16.48 | 18.3 | 3.01 | 84.95 |
| **Ring** | 4.64 | 5.82 | 26.6 | 3.83 | 226.70 |
| **Satimage** | 12.22 | 14.22 | 20.4 | 11.06 | 345.82 |
| **Segment** | 4.55 | 6.45 | 22.2 | 6.69 | 146.18 |
| **Texture** | 6.50 | 7.75 | 25.8 | 14.90 | 352.26 |
| **Twonorm** | 4.42 | 6.06 | 17.4 | 7.40 | 254.88 |

**Table 3**. Results for instance selection with the stratified strategy

| Dataset | Training error | Test error | Number of rules | Rule length | Time(minutes) |
|---|---|---|---|---|---|
| **Magic** | 14.98 | 15.23 | 10.7 | 4.77 | 121.19 |
| **Page-blocks** | 3.59 | 3.83 | 7.8 | 4.94 | 20.73 |
| **Penbased** | 3.18 | 3.87 | 31.3 | 6.42 | 65.36 |
| **Phoneme** | 16.29 | 16.77 | 12.5 | 3.15 | 15.45 |
| **Ring** | 3.38 | 4.86 | 30.0 | 4.12 | 54.08 |
| **Satimage** | 11.28 | 13.05 | 27.9 | 6.72 | 96.31 |
| **Segment** | 3.01 | 5.13 | 25.1 | 6.32 | 39.21 |
| **Texture** | 3.34 | 4.31 | 28.1 | 11.76 | 97.47 |
| **Twonorm** | 3.07 | 4.66 | 18.7 | 7.45 | 58.92 |

**Table 4**. Difference between Table 2 and Table 3, positive is better

| Dataset | Training error | Test error | Number of rules | Rule length | Time ratio, % |
|---|---|---|---|---|---|
| **Magic** | 0.08 | 0.5 | 1.9 | -0.95 | 32.70 |
| **Page-blocks** | -0.07 | 0.13 | 2.3 | -1.45 | 21.94 |
| **Penbased** | 3.88 | 3.59 | -0.8 | -0.19 | 16.97 |
| **Phoneme** | -1.26 | -0.29 | 5.8 | -0.14 | 18.19 |
| **Ring** | 1.26 | 0.96 | -3.4 | -0.29 | 23.86 |
| **Satimage** | 0.94 | 1.17 | -7.45 | 4.34 | 27.85 |
| **Segment** | 1.54 | 1.32 | -2.9 | 0.37 | 26.82 |
| **Texture** | 3.16 | 3.44 | -2.3 | 3.14 | 27.67 |
| **Twonorm** | 1.35 | 1.4 | -1.3 | -0.05 | 23.12 |

**Table 5**. Results for instance selection with the balancing strategy

| Dataset | Training error | Test error | Number of rules | Rule length | Time(minutes) |
|---|---|---|---|---|---|
| **Magic** | 14.62 | 15.08 | 17.3 | 3.63 | 129.65 |
| **Page-blocks** | 2.71 | 3.25 | 18.9 | 4.82 | 18.53 |
| **Penbased** | 3.27 | 3.81 | 30.8 | 6.11 | 91.42 |
| **Phoneme** | 15.63 | 16.88 | 24.0 | 2.84 | 19.62 |
| **Ring** | 3.23 | 5.08 | 30.2 | 3.85 | 68.23 |
| **Satimage** | 10.57 | 12.93 | 27.2 | 5.84 | 85.12 |
| **Segment** | 3.55 | 5.19 | 25.1 | 6.24 | 32.40 |
| **Texture** | 3.37 | 4.45 | 27.0 | 12.81 | 114.79 |
| **Twonorm** | 4.03 | 4.81 | 15.0 | 7.74 | 38.11 |

**Table 6**. Difference between Table 2 and Table 5, positive is better

| Dataset | Training error | Test error | Number of rules | Rule length | Time(minutes) |
|---|---|---|---|---|---|
| **Magic** | 0.44 | 0.65 | -4.7 | 0.19 | 34.98 |
| **Page-blocks** | 0.81 | 0.71 | -8.8 | -1.33 | 19.61 |
| **Penbased** | 3.79 | 3.65 | -0.3 | 0.12 | 23.73 |
| **Phoneme** | -0.60 | -0.40 | -5.75 | 0.17 | 23.10 |
| **Ring** | 1.41 | 0.74 | -3.6 | -0.02 | 30.10 |
| **Satimage** | 1.65 | 1.29 | -6.8 | 5.22 | 24.61 |
| **Segment** | 1.00 | 1.26 | -2.9 | 0.45 | 22.16 |
| **Texture** | 3.13 | 3.30 | -1.2 | 2.09 | 32.59 |
| **Twonorm** | 0.39 | 1.25 | 2.4 | -0.34 | 14.95 |

Applying the balancing strategy changed the behaviour of the algorithm for most of the datasets. For the Phoneme dataset the difference changed from -1.29% to -0.4%. The best improvement was for the Penbased problem, 3.65%. Although the average number of rules increased for 8 datasets out of 9, the average rule length decreased for 6 datasets out of 9, which means that the algorithm has designed more rules, but they are less complex.

The best configurations for the balancing strategy were: Magic: 30%, 200; Page-blocks: 20%, 50; Penbased: 25%, 100; Phoneme: 30%, 50; Ring: 30%, 50; Satimage: 30%, 50; Segment: 25%, 50; Texture: 30%, 50; Twonorm: 20%, 200.

However, the effect of the balanced strategy was mainly demonstrated not in terms of overall accuracy, but in terms of a more balanced classification. To compare how successfully the algorithm recognizes both minority and majority classes, we used the $\text{Recall}_M$ measure from [15]. Table 7 contains the comparison of (1- $\text{Recall}_M$)*100 values for the original algorithm, stratified strategy and balanced strategy.

The presented results prove that using the balanced strategy not only increases the overall classification accuracy, but also allows different classes to be recognized more precisely. However, because for most cases the best results were obtained when using 30% of the sample, applying the balancing strategy to the Page-blocks problem, for example, still resulted in a highly imbalanced subsample. Nevertheless, the improvement for this problem is about 13% compared with original algorithm.

In Tables 8 and 9 we provide the (1- $\text{Recall}_M$)*100 values and accuracy values for the Page-blocks problem for all subsample sizes and all adaptation period lengths.

The lowest (1- $\text{Recall}_M$)*100 value was obtained when the size of the subsample was minimal, i.e. when the subsample is as balanced as possible. We should mention that even for 5% for the Page-blocks datasets it was impossible to create a completely balanced subsample, as the minority class contains only 28 measurements, while all the other classes in this case contained 54 measurements. The bias towards the majority classes

**Table 7**. Comparison of (1- $Recall_M$)*100 values

| **Dataset** | Train. Origin. | Test. Origin. | Train. Stratif. | Test. Stratif. | Train. Balanc. | Test. Balanc. |
|---|---|---|---|---|---|---|
| **Magic** | 18.74 | 19.47 | 18.83 | 19.12 | **17.45** | **17.99** |
| **Page-blocks** | 34.85 | 36.80 | 36.43 | 38.52 | **19.64** | **23.05** |
| **Penbased** | 7.04 | 7.44 | **3.16** | 3.84 | 3.25 | **3.80** |
| **Phoneme** | 18.98 | 20.80 | 21.48 | 22.38 | **16.60** | **18.10** |
| **Ring** | 4.64 | 5.82 | 3.38 | **4.87** | **3.23** | 5.09 |
| **Satimage** | 15.78 | 17.98 | 15.85 | 17.73 | **13.63** | **15.28** |
| **Segment** | 4.55 | 6.45 | **3.01** | **5.13** | 3.55 | 5.19 |
| **Texture** | 6.50 | 7.75 | **3.34** | **4.31** | 3.37 | 4.45 |
| **Twonorm** | 4.42 | 6.06 | **3.07** | **4.66** | 4.03 | 4.81 |

has been almost eliminated thanks to the balancing strategy, as the accuracy value was equal to *7.95*.

For comparison, we provide the accuracy values for the Page-blocks problem.

**Table 8**. (1- $Recall_M$)*100 values for Page-blocks

| **Adaptation period length** | **50** | **100** | **200** | **400** |
|---|---|---|---|---|
| **5%** | **10.18** | **12.92** | 15.18 | **12.06** |
| **10%** | 12.92 | 13.71 | **14.59** | 17.51 |
| **15%** | 20.51 | 19.63 | 18.35 | 19.26 |
| **20%** | 23.05 | 21.23 | 22.72 | 22.36 |
| **25%** | 28.35 | 26.14 | 23.05 | 22.24 |
| **30%** | 28.04 | 27.73 | 28.42 | 29.51 |

**Table 9**. Accuracy values for Page-blocks

| **Adaptation period length** | **50** | **100** | **200** | **400** |
|---|---|---|---|---|
| **5%** | 7.95 | 8.44 | 9.25 | 8.48 |
| **10%** | 5.81 | 6.03 | 6.67 | 7.27 |
| **15%** | 4.00 | 4.50 | 4.90 | 5.77 |
| **20%** | **3.25** | **3.40** | 3.93 | 4.40 |
| **25%** | 3.78 | 3.78 | **3.69** | 4.15 |
| **30%** | 3.38 | 3.69 | 3.78 | **4.06** |

For 5% of the training set in the subset the accuracy decreases by a factor of 2 compared to values obtained when using 30% of the sample. However, the classifiers obtained with 5% of the sample can be more suitable, as they are capable to successfully classify all classes with the same accuracy.

In Tables 10 and 11 we also provide the confusion matrixes for the Page-blocks problem to show the effect of the balancing strategy.

In every row of tables 10 and 11 the predicted class having the largest number of instances classified was marked in bold. The original algorithm correctly classifies the first, majority class, but the other classes are also classified into the first class, except the second one. This means that actually the algorithm correctly classifies only the first and second classes. When using the balancing strategy, the situation changes, and most of the instances are correctly classified in their class, however, this leads to a lower general classification quality.

For the remaining problems similar results were obtained. The main trend is that more balanced classifiers can be trained with a smaller size of the sample. If the sample becomes balanced at some particular percentage, there is no sense in decreasing its size any more.

In Figures 3-10 we show the surface plot as a graphical representation of the dependence of accuracy on the size of the subsample and the length of the adaptation period.

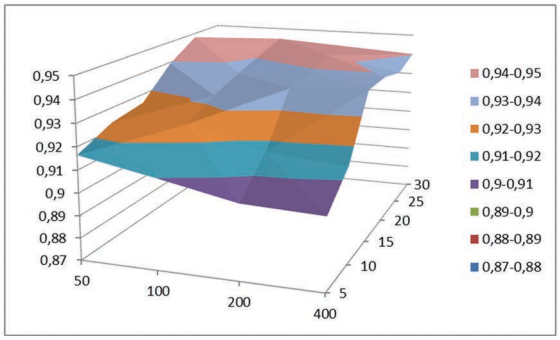


**Figure 3**. Segment

**Table 10**. Confusion matrix for the Page-blocks problem, original algorithm

| Class | Pred. 1 | Pred. 2 | Pred. 3 | Pred. 4 | Pred. 5 | Unknown |
|---|---|---|---|---|---|---|
| True 1 | **487.1** | 2.90 | 0 | 0.36 | 0.90 | 0 |
| True 2 | 3.09 | **29.0** | 0.18 | 0.18 | 0.45 | 0 |
| True 3 | **1.72** | 0 | 1.09 | 0 | 0 | 0 |
| True 4 | **2.45** | 0.09 | 0 | 6 | 0.18 | 0 |
| True 5 | **8.73** | 0 | 0.27 | 0 | 2.54 | 0 |

**Table 11**. Confusion matrix for the Page-blocks problem, balanced strategy

| Class | Pred. 1 | Pred. 2 | Pred. 3 | Pred. 4 | Pred. 5 | Unknown |
|---|---|---|---|---|---|---|
| True 1 | **452.0** | 19.36 | 3.72 | 6.54 | 9.64 | 0.09 |
| True 2 | 1.72 | **30**.27 | 0.09 | 0.36 | 0.36 | 0.09 |
| True 3 | 0.18 | 0 | **2.54** | 0.09 | 0 | 0 |
| True 4 | 0 | 0.27 | 0 | **8.27** | 0.18 | 0 |
| True 5 | 1.27 | 0.36 | 0.18 | 0.36 | **9.36** | 0 |

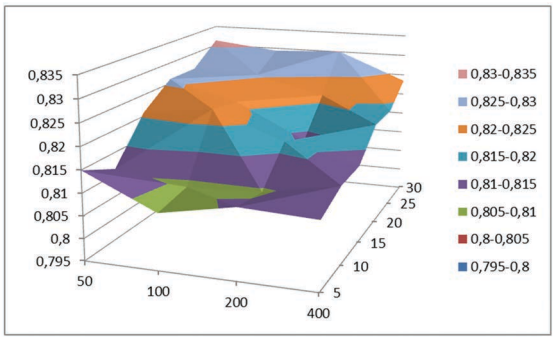


**Figure 4**. Phoneme

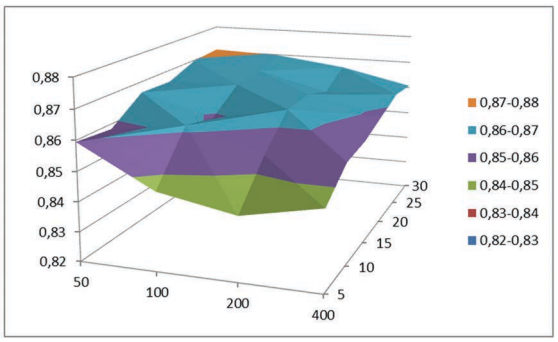


**Figure 6**. Satimage

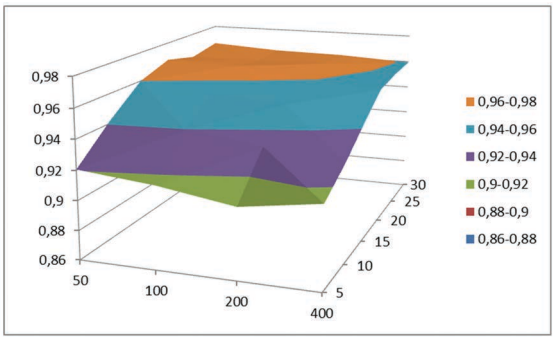


**Figure 5**. Page-blocks

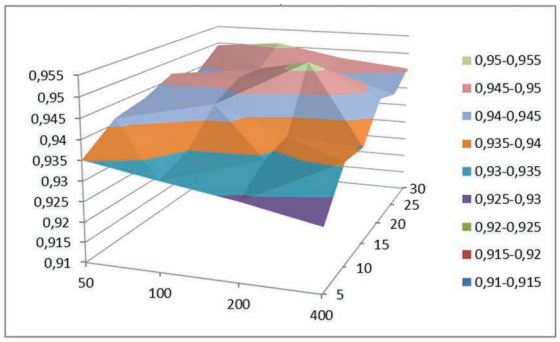


**Figure 7**. Twonorm

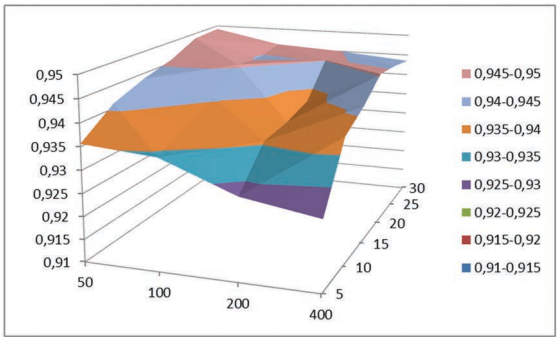


**Figure 8**. Ring

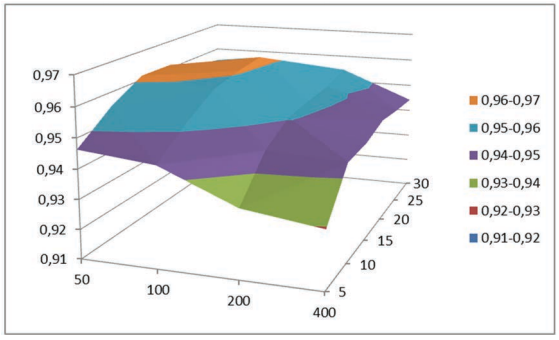


**Figure 9**. Penbased

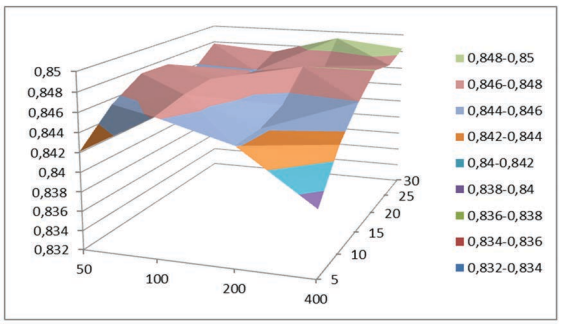


**Figure 10**. Magic

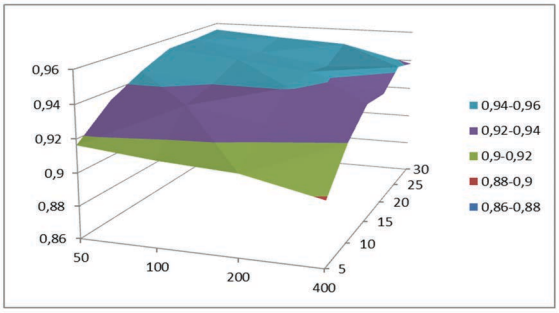


**Figure 11**. Texture

In most of the cases the increasing of the subsample size leads to better classification accuracy. The dependence on the adaptation period length is not so straightforward; however for most of the problems shorter adaptation periods are more preferable. This happens because for most of the problems the population is capable of adjusting to the new subsample within 50 generations. Moreover, a shorter adaptation period gives the possibility to update the $U_i$ values more often and better distinguish the problematic areas of the feature space.

To compare the time required for computation by the original algorithm and the instance selection method, the computation time was averaged over all adaptation period lengths for every subsample size. The resulting time values were compared to the ones for the original algorithm. Next, the acceleration rate equal to $T_{orig}/T_{IS}$ was calculated. Table 12 contains the acceleration values.

The maximal acceleration value achieved is equal to 21.5, and the minimal to 3.09. For most of the problems the algorithm with adaptive instance selection had results not worse than the original method already at the point of 10-15% of the training sample being used, which means that at the same level of classification quality the instance selection allows the algorithm to work 6-12 times faster.

For better understanding of the training process with instance selection, we provide an error graph for the best individual for the overall sample and the subsample. In Figure 11 the graph for the Penbased problem with an adaptation period of 50 and subsample size of 15% is presented. In Figure 12 the graph for the Penbased problem with an adaptation period of 400 generations and subsample size of 15% is presented.

One of the characteristics of the proposed adaptive instance selection algorithm is that the subsample error rate appears to be larger than the overall training error. Nevertheless, this does not prevent a successful training process. The reason for such behaviour is that the instance selection focuses on instances which are difficult to classify (most of the time they are instances on the border between classes) and includes them into the training subset. In the first adaptation periods, the subsample error and the overall error are very similar, which means that the measurements are selected almost uniformly. But later the situation changes, because instances which are easy to classify get larger

**Table 12**. Time acceleration values

| Dataset | 5 | 10 | 15 | 20 | 25 | 30 | Orig |
|---|---|---|---|---|---|---|---|
| **Magic** | 13.51 | 9.11 | 6.15 | 4.83 | 3.47 | 3.09 | 1 |
| **Page-blocks** | 13.51 | 8.90 | 6.70 | 5.41 | 4.41 | 3.60 | 1 |
| **Penbased** | 14.04 | 9.47 | 6.39 | 5.02 | 3.61 | 3.20 | 1 |
| **Phoneme** | 21.50 | 11.39 | 8.72 | 6.05 | 4.96 | 4.16 | 1 |
| **Ring** | 16.13 | 10.47 | 6.83 | 5.57 | 4.22 | 3.46 | 1 |
| **Satimage** | 17.25 | 11.62 | 8.10 | 5.73 | 4.34 | 3.50 | 1 |
| **Segment** | 19.65 | 10.65 | 7.37 | 5.56 | 4.42 | 3.72 | 1 |
| **Texture** | 17.70 | 11.09 | 7.15 | 4.82 | 3.76 | 3.15 | 1 |
| **Twonorm** | 18.54 | 12.41 | 8.86 | 6.72 | 5.13 | 4.39 | 1 |

counter values and lower probabilities to be chosen. As these instances are still present in the training sample and classified correctly, the overall training error becomes lower than the subsample error.

When increasing the adaptation period length, the training process becomes slower. In Figure 12 it is seen that at the first adaptation period the training subsample error becomes lower than the overall training set error. This happens because the population overfits to the subsample, i.e. the best rule base for the subsample describes the trends which are presented only in the subsample. During the several next adaptation periods the error on the subsample and training set is almost identical, but then the probabilities are changed so that the algorithm focuses on problematic areas of the search space, which leads to a larger subsample error. In the case of a longer adaptation period, the difference between the training set and subset errors is smaller than for the shorter adaptation period.

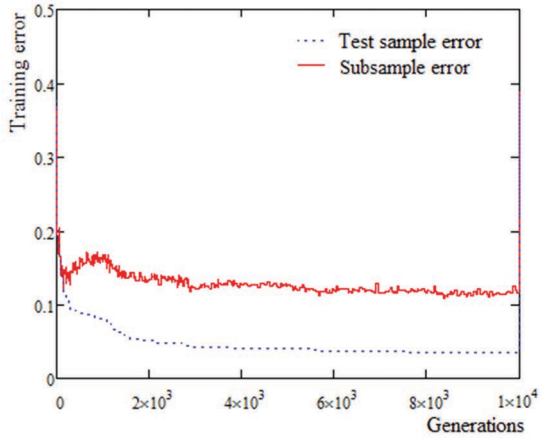


**Figure 12**. Accuracy change during the training process, Penbased problem, 50 generations

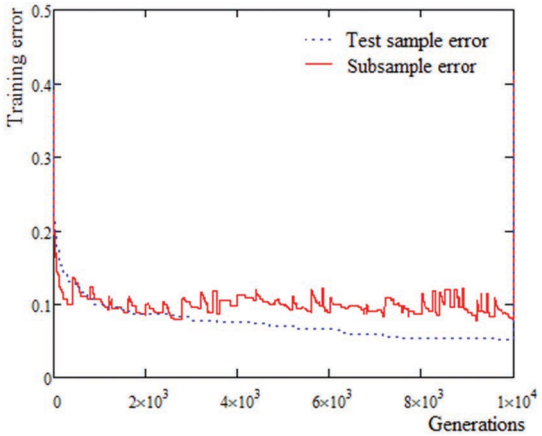


**Figure 13**. Accuracy change during the training process, Penbased problem, 400 generations

Let us consider the training process in the sense of different classification quality measures. The Page-blocks problem will be considered, as it is the most imbalanced problem. To calculate the quality measures such as Accuracy (overall classification accuracy), $\text{Precision}_\mu$, Recall, Fscore, $\text{Precision}_M$, $\text{Recall}_M$, $\text{Fscore}_M$ taken from [15], the confusion matrix was calculated for every generation. In Figure 13 we provide the averaged graphs for all measures averaged over all algorithm runs.

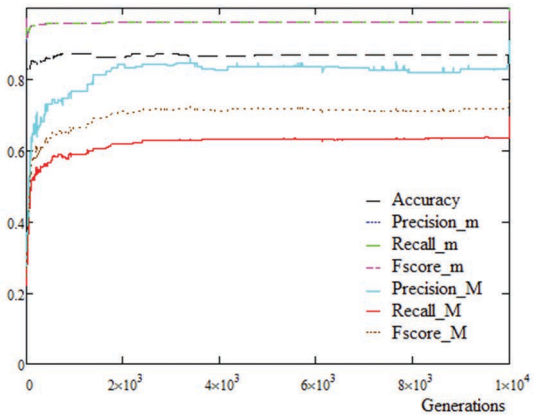


**Figure 14**. The change in classification quality, Page-blocks problem, original algorithm

For this problem the $\text{Precision}_{\mu}$, Recall and Fscore measures are almost identical, but $\text{Precision}_M$, $\text{Recall}_M$, $\text{Fscore}_M$ are different. This happens due to the fact that the last measures are macro-measures and are calculated for each class separately and then averaged. In the case of imbalanced classification, the $\text{Precision}_M$ becomes larger than Accuracy, but $\text{Recall}_M$ becomes smaller. The errors for each of the five classes are presented in Figure 14.

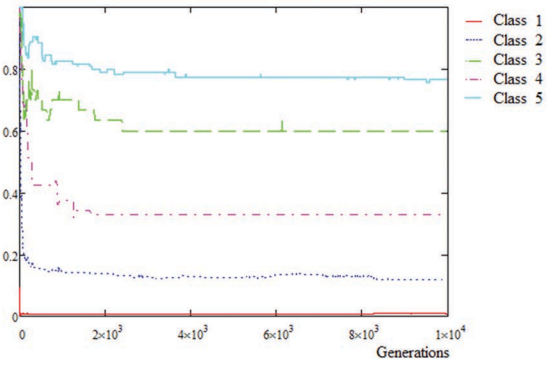


**Figure 15**. Classification errors for five classes, Page-blocks problem, original algorithm

Only the first, most represented class can be recognized correctly by the original algorithm – the error values for the remaining classes are high or very high. The next two Figures, 15 and 16, show the change in classification quality measures and classification errors for the instance selection algorithm with the balancing strategy, with the following parameters: 5% of the training sample in the subsample and an adaptation period of 50 generations.

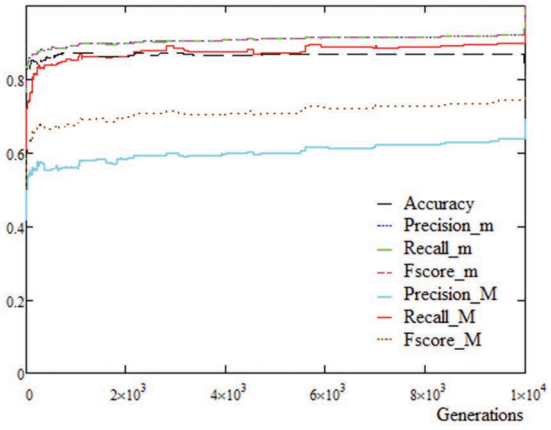


**Figure 16**. The change in classification quality, Page-blocks problem, balanced instance selection

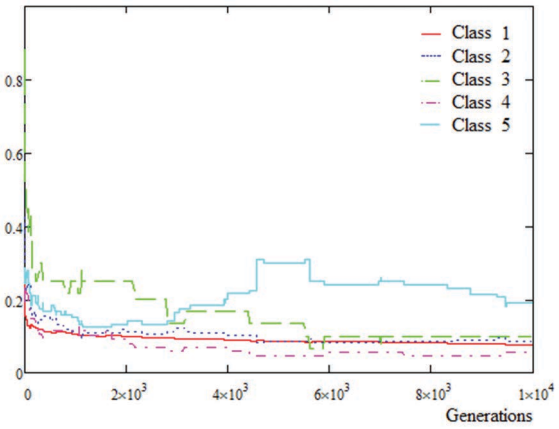


**Figure 17**. Classification errors for five classes, Page-blocks problem, balanced instance selection

Compared to the previous case, the solutions are now more balanced, which is especially noticeable in Figure 16. Here all classes have error rates of about 0.1 except for the fifth class, which has 0.2. Also, the $\text{Fscore}_M$ values are larger than for the original algorithm. The presented results prove that the instance selection method allows the classification quality to be increased when using the balanced strategy for imbalanced datasets.

For comparison with other methods, an additional computation experiment was performed on the same set of problems. The algorithm was tested without instance selection, but with larger resources, i.e. the population size was 210, the number of generations was 50000, and the number of rules did not change and was equal to 40. Several classification algorithms were chosen for comparison from [16, 17]. Among them there was also the prototype algorithm developed by the H. Ishibuchi

**Table 13**. Comparison with other approaches

| **Dataset** | HEFCA IS | HEFCA Orig. | HEFCA Res. | Fuzzy GBML [4] | Paral. Fuzzy GBML | FARC-HD [16] | Bio HEL [17] |
|---|---|---|---|---|---|---|---|
| **Magic** | 15.08 | 15.73 | 15.87 | 15.42 | **14.89** | 15.49 | - |
| **Page-blocks** | **3.25** | 3.96 | 3.78 | 3.81 | 3.62 | 4.99 | - |
| **Penbased** | 3.81 | 7.46 | 5.02 | **3.07** | 3.30 | 3.96 | 6.00 |
| **Phoneme** | 16.88 | 16.48 | **15.36** | 15.43 | 15.96 | 17.86 | - |
| **Ring** | **5.08** | 5.82 | 5.52 | 6.73 | 5.25 | 5.92 | - |
| **Satimage** | 12.93 | 14.22 | 12.86 | 15.54 | 12.96 | 12.68 | **11.60** |
| **Segment** | 5.19 | 6.45 | 5.62 | 5.99 | 5.90 | - | **2.90** |
| **Texture** | **4.45** | 7.75 | 4.90 | 4.64 | 4.77 | 7.11 | - |
| **Twonorm** | 4.81 | 6.06 | 5.57 | 7.36 | **3.39** | 4.72 | - |

group, and also their parallel implementation with the training subsample rotation.

The presented approach with the balancing strategy, noted as HEFCA IS (Hybrid Evolutionary Fuzzy Classification Algorithm with Instance Selection) appeared to be the best in the sense of accuracy for 3 problems out of 9. Moreover, this algorithm has overcome not only the original method (HEFCA Orig.) on 8 problems out of 9, but also the method with 10.5 times larger resource (HEFCA Res.) on 7 problems. The prototype algorithm and its parallel implementation were outperformed on 4 problems out of 9. On the Satimage problem the results are almost identical. The parallel implementation with subsamples rotation used 10.5 times more computational resource and 7 threads.

Table 14 contains the comparison of the time required for training by the algorithm with instance selection and the Parallel fuzzy GBML [4].

**Table 14**. Training time comparison

| **Dataset** | HEFCA IS | Parallel fuzzy GBML | Ratio |
|---|---|---|---|
| **Magic** | 129.65 min. | 22.58 min. | 5.74 |
| **Page-blocks** | 18.53 min. | 4.74 min. | 3.91 |
| **Penbased** | 91.42 min. | 35.56 min. | 2.57 |
| **Phoneme** | 19.62 min. | 13.19 min. | 1.49 |
| **Ring** | 68.23 min. | 22.52 min. | 3.03 |
| **Satimage** | 85.12 min. | 15.38 min. | 5.53 |
| **Segment** | 32.40 min. | 4.69 min. | 6.91 |
| **Texture** | 114.79 min. | 15.72 min. | 7.30 |
| **Twonorm** | 38.11 min. | 7.84 min. | 4.86 |

On average, the presented approach is 4.6 times slower, but the parallel algorithm used 7 threads for calculation, allowing it to train up to 7 times faster. One should mention that the presented time values for instance selection are for the best configurations, and for most of the problems the best results were obtained when using 30% of the training sample. This means that the time required for training can be decreased even more without a significant loss in classification accuracy.

## 5 Conclusion

The proposed adaptive instance selection algorithm for imbalanced datasets creates samples of a fixed size out of the original training sample to guide the training process by taking the information about classification results into consideration. The goal of this method is to decrease the time required for computation along while increasing the quality of classification. The testing results prove that the proposed method allows a significant improvement in classification quality, especially for imbalanced datasets because of the use of the balancing strategy.

The resulting classifiers are capable of recognizing all classes with similar accuracy values, which is important for many real-world problems. The time required for training decreases significantly, and can be adjusted by changing the size of the subsample. The resulting accuracy is at the same level as the best algorithms for the presented set of problems, or even better.

The proposed adaptive instance selection method is universal and can be applied to any other iteration-based machine learning technique used for classification, including non-evolutionary algorithms.

## References

[1] A. Fernandez, S. Garcia, J. Luengo, E. Bernado-Mansilla, F. Herrera, Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy, and Comparative Study, Evolutionary Computation, IEEE Transactions on (Volume:14, Issue: 6), June 21, 2010, pp. 913 – 941.

[2] L. B. Booker, D. E. Goldberg, and J. H. Holland, Classifier systems and genetic algorithms, Artif. Intell., vol. 40, no. 1–3, Sep. 1989, pp. 235–282.

[3] Bodenhofer U., Herrera F. Ten Lectures on Genetic Fuzzy Systems, Preprints of the International Summer School: Advanced Control—Fuzzy, Neural, Genetic. – Slovak Technical University, Bratislava., 1997. p. 1–69.

[4] Ishibuchi H., Mihara S., Nojima Y. Parallel Distributed Hybrid Fuzzy GBML Models With Rule Set Migration and Training Data Rotation, IEEE Transactions on fuzzy systems, vol. 21, n. 2., April 2013.

[5] Ishibuchi H., T. Yamamoto, Rule weight specification in fuzzy rule-based classification systems, IEEE Trans. Fuzzy Systems 13, 2005, pp. 428–435.

[6] E. Semenkin, M. Semenkina, Self-configuring genetic algorithm with modified uniform crossover operator, in Y. Tan, Y. Shi, Z. Ji (Eds.), Advances in Swarm Intelligence, PT1, LNCS 7331, 2012, pp. 414-421.

[7] E. Semenkin, M. Semenkina, Self-Configuring Genetic Programming Algorithm with Modified Uniform Crossover, in Proc. of the IEEE Congress on Evolutionary Computation (CEC 2012), Brisane (Australia), pp. 1-6, 2012.

[8] M. Semenkina, E. Semenkin, Hybrid self-configuring evolutionary algorithm for automated design of fuzzy classifier, in Y. Tan, Y. Shi, C.A.C. Coello (Eds.), Advances in Swarm Intelligence, PT1, LNCS 8794, 2014, pp. 310-317.

[9] J. R. Cano, F. Herrera, M. Lozano, Stratification for scaling up evolutionary prototype selection, Pattern Recognition Letters, 2004 Volume 26, Issue 7, 15 May 2005, Pages 953–963.

[10] J. R. Cano, F. Herrera, M. Lozano, A Study on the Combination of Evolutionary Algorithms and Stratified Strategies for Training Set Selection in Data Mining, Advances in Soft Computing Volume 32, 2005, pp 271-284.

[11] A. Fernndez, M. J. Jesus, F. Herrera, Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets, International Journal of Approximate Reasoning, 2009, pp. 561–577.

[12] A. Fernndez, S. Garca, M. J. Jesusb, F. Herrera, A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets, Fuzzy Sets and Systems 159, 2008, pp. 2378 – 2398.

[13] Asuncion A., Newman D. UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences, 2007.

[14] J. Alcal-Fdez, L. Snchez, S. Garcia, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernndez, and F. Herrera, KEEL: A software tool to assess evolutionary algorithms for data mining problems, Soft Comput., vol. 13, no. 3, pp. 307–318, Feb. 2009.

[15] Sokolova M., Lapalme G. A systematic analysis of performance measures for classification tasks. – Information Processing and Management 45, 2009, pp. 427–437.

[16] J. Alcala-Fdez, R. Alcala, F. Herrera, A fuzzy association rulebased classification model for high-dimensional problems with genetic rule selection and lateral tuning, IEEE Trans. Fuzzy Syst., vol. 19, no. 5, Oct. 2011, pp. 857–872.

[17] J. Bacardit, E. K. Burke, N. Krasnogor, Improving the scalability of rule-based evolutionary learning, Memetic Comput. J., vol. 1, no. 1, Mar. 2009, pp. 55–67.

**Vladimir Stanovov** received the Bachelor of Science and Master of Science degrees in systems analysis and control from Reshetnev Siberian State Aerospace University, Krasnoyarsk, Russia, in 2012 and 2014 respectively. His research interests include genetic fuzzy systems, self-configured evolutionary algorithms and machine learning. He is currently a PhD student at the Siberian State Aerospace University. Mr. Stanovov received the Best Student Paper Award from the 4th International Congress on Advanced Applied Informatics in 2015.

**Eugene Semenkin** received his Master in Applied Mathematics degree from Kemerovo State University (Kemerovo, USSR) in 1982, his PhD in Computer Science from Leningrad State University (Leningrad, USSR) in 1989 and his DSc in Engineering and Habilitation from the Siberian State Aerospace University (Krasnoyarsk, Russia) in 1997. Since 1997, he has been a professor of systems analysis at the Institute of Computer Science and Telecommunications of the Siberian State Aerospace University. His areas of research include the modelling and optimization of complex systems, computational intelligence and data mining. He has been awarded the Tsiolkovsky Badge of Honour by the Russian Federal Space Agency and the Reshetnev medal by the Russian Federation of Cosmonautics.

**Olga Semenkina** received her Master in Mathematics degree from Kemerovo State University (Kemerovo, USSR) in 1982, her PhD in Computer Science from Siberian Aerospace Academy (Krasnoyarsk, Russia) in 1995 and her DSc in Engineering from the Siberian State Aerospace University (Krasnoyarsk, Russia) in 2002. Since 2003, she has been a professor of Higher Mathematics of the Siberian State Aerospace University. Her areas of research include the modelling of complex systems and discrete optimization. She has been awarded the Badge of Honoured Worker of Higher Education of the Russian Federation.