Obtaining Pareto Front in Instance Selection with Ensembles and Populations

Mirosław Kordos[1(✉)], Marcin Wydrzyński[1], and Krystian Łapa[2]

[1] Department of Computer Science and Automatics, University of Bielsko-Biala, Bielsko-Biała, Poland
mkordos@ath.bielsko.pl
[2] Częstochowa University of Technology, Institute of Computational Intelligence, Częstochowa, Poland
krystian.lapa@iisi.pcz.pl

Abstract. Collective computational intelligence can be used in several ways, for example as taking the decision together by some form of a bagging ensemble or as finding the solutions by multi-objective evolutionary algorithms. In this paper we examine and compare the application of the two approaches to instance selection for creating the Pareto front of the selected subsets, where the two objectives are classification accuracy and data size reduction. As the bagging ensemble members we use DROP5 algorithms. The evolutionary algorithm is based on NSGA-II. The findings are that the evolutionary approach is faster (contrary to the popular belief) and usually provides better quality solutions, with some exceptions, were the outcome of the DROP5 ensemble is better.

1 Introduction

Data preprocessing is frequently the most important step in data mining, even more important than choice of the classifier and its parameters, as even the best classifier cannot produce good outcome if the data quality is poor. One part of data preprocessing is data selection. Improving the data quality is one objective of data selection. The other objective is limitation the data size to make the data easier to interpret and analyze and to make the learning process faster. The data selection process can be seen as a two-objective optimization, aimed at data size reduction and classification accuracy improvement. Frequently it happens that a small reduction of data size (when done properly) can achieve both objectives. Although further reduction of data size also reduces the prediction accuracy, it can still be beneficial for data analysis and fast experiments with the learning model.

In this work we want to find a Pareto-front in the compression-accuracy space - that is such a set of solutions, that for any of them there does not exist a solution that can simultaneously improve both: compression and accuracy. The closer is the Pareto front situated to the 0% of selected instances and 100% accuracy point in the accuracy-compression space (upper-left corner in Figs. 2,

L. Rutkowski et al. (Eds.): ICAISC 2018, LNAI 10841, pp. 438–448, 2018.
https://doi.org/10.1007/978-3-319-91253-0_41

4, 5, and 6.) the better. Data selection can be decomposed into feature and instance selection and in our previous work [1] we analyzed the relations and interactions between feature and instance selection. In this work we focus on instance selection and its Pareto-Front.

The first (obvious) observation is that in order to achieve the front we need many solutions. On the other hand most instance selection methods produce only a single solution. There are two ways to obtain the Pareto front. One is to perform the instance selection multiple times with different parameters of the instance selection method, that however can be applied only to these instance selection methods that can be parameterized in this way. The other solution is to use ensembles or populations of instance selection algorithms [2,3,9]. Ensemble methods are widely used in classification and regression problems as they are known to obtain better results than the single best method. The same can be applied to instance selection and the same methods of differentiating the results can be applied (as instance bagging or feature bagging for example) and different voting schemes can be used to move along the Pareto front, as discussed in Sect. 2.

Evolutionary algorithms in contrast use populations and cooperation between the population members (using the crossover operator). There are also known multi-objective evolutionary algorithms, as the NSGA-II [4], which provides additional operators to ensure that the best members of the population occupy possibly uniformly the Pareto front. Using them is simply faster than running many single-objective optimizations with different parameters and it may also produce better results (lower Pareto front), as discussed in Sect. 3.

In Sect. 4 we experimentally compare the Pareto front obtained by the two approaches: a bagging ensemble of DROP5 algorithms and an NGSA-II based evolutionary instance selection.

2 Bagging Ensembles of DROP5 Instance Selection Algorithms

The ensemble methods are proved to be efficient for enhancing the classification process and many solutions are widely used for that purpose [5].

However, the idea of ensembles of instance selection algorithms has been considered only by few authors. The first idea related to ensembles was presented in the all-kNN algorithm, where the several ENN algorithms with different k values were used [6]. In [7] the authors attempted to adapt boosting to instance selection, where the objective was not only to improve the success rate, but also to reduce the data size. The weight given by boosting to each instance defined relevance of the instance, and a statistical test was used to decide whether it can be discarded without affecting classification accuracy.

In [8] classifier ensembles were constructed using weighted instance selection. In [3,9] two variants of bagging of instance selection algorithms were discussed; in the first paper feature based bagging and in the second instance-based bagging. In [10] the authors applied boosting for several instance selection algorithms, as CNN, IB3, DROP3, ICF, approaching the instance selection problem

as a two-class classification. They tested several boosting methods, as AdaBoost, FloatBoost, MultiBoost, ReweightBoost and the obtained results confirmed that in most cases the boosting methods showed better performance than the single instance selection algorithm.

The boosting methods, should be used only after applying noise filter or on data that does not require noise filters. Otherwise, the noisy instances would be promoted by the algorithm, as they are usually mis-classified, what can lead to incorrect results. For the same reason in the DROP (Decremental Reduction Optimization Procedure) algorithm noise filter is applied before the condensation part [11].

The ENN (Edited Nearest Neighbor) [11] algorithm is a noise filer. It starts from a whole dataset and check which instances are wrongly classified by k-NN. Each wrongly classified instance is marked for removal. Finally all the instances marked for removal get removed.

The instance selection algorithms from the DROP family methods belongs to the best instance selection methods for classification tasks [12,13]. Two concepts are defined in those methods: nearest neighbors and associates. Nearest neighbors of an instance p are those k instances to which the distance from the instance p is shorter than to the remaining instances. Associates are those instances that have p as one of their k nearest neighbors.

The DROP5 algorithm used in this work was developed based on its predecessors in the following way:

- DROP1 eliminates an instance p, if this does not affect the classification of its associates.
- DROP2 before starting the selection sorts the instances in descending order by the distances from the instance to its nearest enemy (the instance from another class). So first the instances located among the same class instances are processed and later the instances close to the class boundary.
- DROP3 additionally first applies a noise filter that works like the ENN algorithm, removing the instances incorrectly classified by k-NN.
- DROP4 applies a modified version of the noise filter, by additionally verifying if removing an instance does not cause a misclassification of another instance and if it does, the instance will not be removed.
- DROP5 is similar to DROP2, but instead of using the noise filter, it starts the analysis from the instances that are closest to the nearest enemies (those on the class boundary). In this way as a matter of fact the noise filter is applied at the first stage. However, its computational complexity is still $O(n^3)$.

To obtain the Pareto front we used an ensemble of 10 instance selection algorithms, were the differentiation between the ensemble members was achieved by providing to each member a random subset of the data set. Then we analyzed how many members of the ensemble voted for each instance and thus we obtained 10 datasets: the first one containing the instances for which every member voted (however, this was frequently an empty set), the next one for which at least 9 members voted, and so one. The last subset was the biggest and contained the instances for which at least one member voted. We evaluated the classification

accuracy for classifiers trained on each of the datasets on the test set. In this way we obtained the Pareto front in the compression-accuracy space. That allows us to select one point on the front depending on the importance we assign to the compression and classification accuracy. The results are presented in the Sect. 4.

3 NSGA-II Based Evolutionary Instance Selection

The way in which standard genetic or evolutionary algorithms work is with very high probability well known to the reader so we will not waste space explaining this, especially that all the details can be easily found in literature [4,14–16].

There are two fundamental differences between the classical and evolutionary-based instance selection algorithms. In the first case a single solution is obtained and some properties of the search must be a priori defined, as for example how to tell the boundary point from the point situated among the same class members, how to define a noisy point and so on. A great advantage of evolutionary-based instance selection is that we do not have to carry about all the definitions and parameters.

Each individual in the genetic population encodes the entire training set and the value at each chromosome position indicates whether the instance is present (value > 0) or not (value $= 0$). In case of instance selection the first value $= 1$. In case of instance weighting it can be a real number between 0 and 1. Most of evolutionary approaches to instance selection define the fitness function in a similar way, as a weighted sum of the achieved compression and classification accuracy on the test set. Thus to obtain a set of solutions the optimization has to be performed several times with various weights. However, this can be avoided using a multi-objective evolutionary algorithms [4,16], as we use in the experimental section.

There have been already some propositions in the literature to use genetic or evolutionary algorithms for instance selections [17–22]. The results obtained by these authors were better in terms of accuracy-compression balance than the results of the best classical algorithms as DROP3, DROP4 and DROP5, however they compared only single points and not the whole Pareto fronts.

It is widely believed that evolutionary optimization can find better solutions that the local search or gradient based-solutions, but its computational cost is much higher. As the first statement is generally true, we show in this work that in the case of instance selection the second statement can be false. The evolutionary instance selection method we use can find the solution in $O(n^2)$ time, while the best classical methods, as the DROP family has the $O(n^3)$ complexity, which for large datasets can be prohibitive. Thus in that case evolutionary methods can be better at both: the obtained results and the evaluation time. We calculate and sort the distance matrix only once at the beginning of the optimization and then reading the classes of the nearest neighbors from the sorted arrays is very fast.

Because the purpose of this work is to obtain not a single solution, but a Pareto front that consists of the set of the best solutions (called also non-dominated solutions), we use a multi-objective evolutionary algorithm based

on NSGA-II [4], as shown in the pseudo-code. The two criteria that we use are compression and classification accuracy. We tested 3 different solutions: based on single-objective evolutionary algorithm with changing the weights for accuracy and compression in the fitness function, the SEEA-based solution [4] and the NSGA-II based solution. The last one in most cases produced the best results, so we decided to use it in the final experiments.

Although there are newer multi-objective evolutionary algorithms, NSGA-II often produces the best or at equal results, and due to the fact that the NSGA-II method is well known, we have decided to use it and modify it for the purpose of this study, by adjusting it to binary instance weights, introducing the proper population initialization (the probability of zero is set to the expected compression level) [23] and additional parameters that force the small values to zeros (which rejects the not very useful instances). In [24] the authors investigated the concept of e-dominance with multi-objective evolutionary algorithms and found out that in some cases this concept significantly helps to reduce the execution time. However, e-dominance can also drastically slow down the process for problems where the number of objective vectors is small. We did not use e-dominance, because in our case it did not improved the performance.

We can perform either binary instance selection or instance weighting. With instance weighting each instance is assigned a value between 0 and 1 representing the importance of this instance. While calculating the majority class of an instance k-NN adds the class of its neighbors multiplied by their weights. To calculate the compression we treat as rejected only those instances with zero weights and all others as selected. If during the optimization a given weight in a given individual takes a value below some threshold (e.g. 0.01) it is forced to zero to ensure effective instance rejection. The instance weighting method allows for achieving higher accuracy in the regions of very stronger compression. However, instance weighting is mostly useful for the instance selection process. In the final prediction all the non-zero values can be converted to ones usually with little loss of accuracy. But if we are focused more or accuracy improvement then binary values used during the optimization are usually producing better solutions with low compression but high prediction accuracy, frequently even above the accuracy obtained on the entire dataset (with the exception of situations as in Fig. 6). There is much more to study in the area of instance selection vs instance weighting and the detailed analysis will be conducted in our future research.

4 Experiments and Results

We conducted the experiments on classification datasets from the KEEL Repository [25]. The experiments with DROP5 were performed in RapidMiner using the Data Selection Extension [3,9] with the Weka Instance Selection Module [26], which contained the DROP5 algorithm. The experiments with evolutionary instance selection were performed using a software we created in C# language. We make available all the software, including GPU-based implementation (with

Algorithm 1. NSGA-II

```
1: P := initialization(N, M) //P is population, F is Pareto front
2: evaluation(P)
3: F = fast_nondominated_sort(P, N)
4: crowding_distance(F)
5: while stop_condition() do
6:    P' = ∅
7:    for i = 1 to N do
8:       parentA = select_parent(P)
9:       parentB = select_parent(P)
10:      child = new_individual(parentA, parentB)
11:      P' = P' ∪ child
12:   end for
13:   evaluation(P')
14:   P = P ∪ P'
15:   F = fast_nondominated_sort(P, 2N)
16:   crowding_distance(F)
17:   P = selection(P, F)
18: end while
19: return F_1 {list of non dominated individuals}
```

Nvidia Cuda Toolkit) of evolutionary instance selection and the detailed results of all the experiments so that the interested reader can find much more information at our web page *www.kordos.com/icaisc*2018.

Although the computational time between different implementations could not be compared directly, as the RadidMiner implementations are generally much slower due to several reasons, its dependence on the number of instances can be. In the first case it was almost $O(n^3)$ what agrees with the theoretical complexity of the DROP algorithms and in the second case it was about $O(nlog(n))$, although we are aware that for a datasets much bigger than that used in the experiments it may asymptotically approach $O(n^2)$, which is the complexity of calculating the distance matrix for k-NN. Of course the time can be reduced in both cases by using more advanced approximate calculations of the distance matrix, but this will not change the relation.

All the tests were run in 10-fold crossvalidation. However, first we randomly changed the order of the instances in each dataset and then we used linear sampling in crossvalidation to ensure that both of the methods use exactly the same training and test subsets - so that the comparison is made on exactly the same data. DROP5 produces the same results with each run on the same data. The NSGA-II based optimization while repeated on the same data displays some variability, but mostly in the number of selected instances and the variability of the obtained accuracy is really very low, as can be seen from the figures how closely particular points adhere to the blue line. The accuracy obtained on the whole dataset is shown in the green line. The final classification algorithm and the inner evaluation algorithm was k-NN with optimal k for each dataset (the k allowing for the highest classification accuracy). Also the same k value

was used for DROP5. The optimal k was usually 1 for the datasets, where the achievable classification accuracy was about 99% and growing with the decrease of the accuracy, reaching 7 or more for the accuracies below 80%. However, in the experiments, we limited the maximum k to 7. For the NSGA-II based optimization we used $S = 96$ individuals and from $E = 20$ to 200 epochs - more for larger datasets.

We also trained decision trees of the type described in [27] and the MLP neural networks with the VSS algorithm [28] on the selected subsets to verify how other classifiers would work on the data and the classification accuracies were highly correlated to that obtained with the optimal k k-NN. We have also observed some interesting issues about the optimization of instance selection process for particular final classifier models. However, formulating the conclusions and recommendations require further studies, which will be another topic of our future research (Fig. 1).

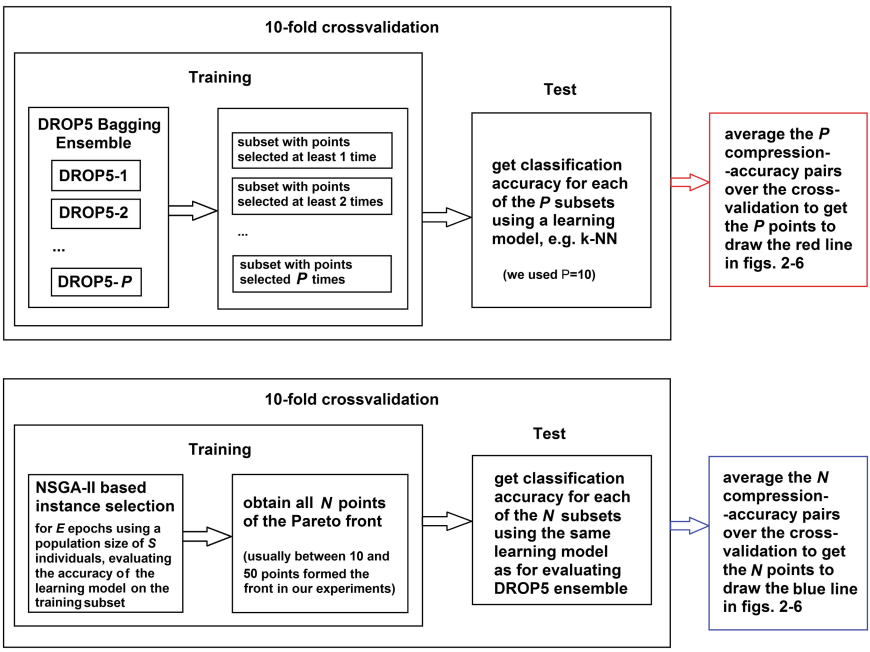

Fig. 1. The experimental setup

5 Conclusions

We discussed the problem of obtaining a Pareto front in the accuracy-compression space in instance selection. We evaluated two approaches: bagging

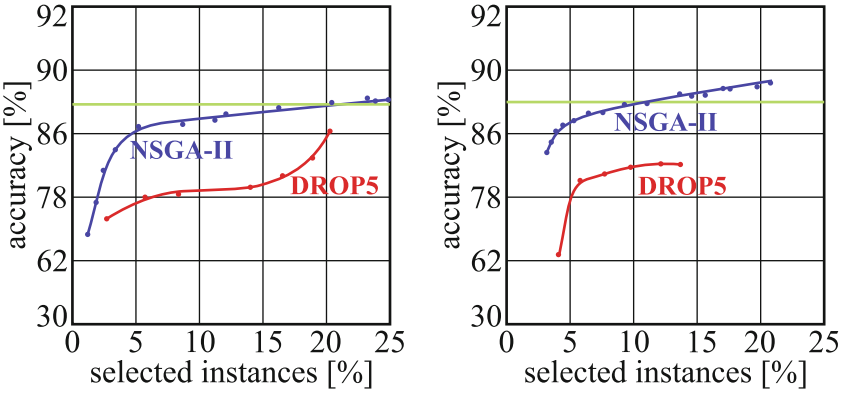

Fig. 2. The Pareto fronts for Balance (left) and Ionosphere (right) datasets. Light green: accuracy without instance selection. (Color figure online)

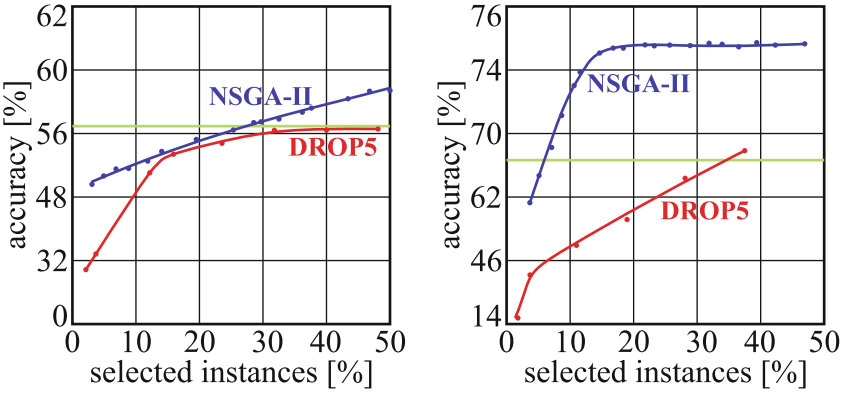

Fig. 3. The Pareto fronts for Yeast (left) and Led7digit (right) datasets. (Color figure online)

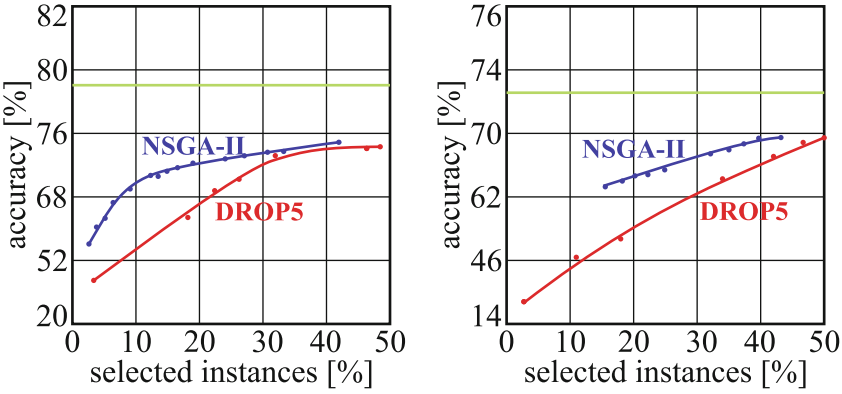

Fig. 4. The Pareto fronts for Bupa (left) and Vehicle (right) datasets. (Color figure online)

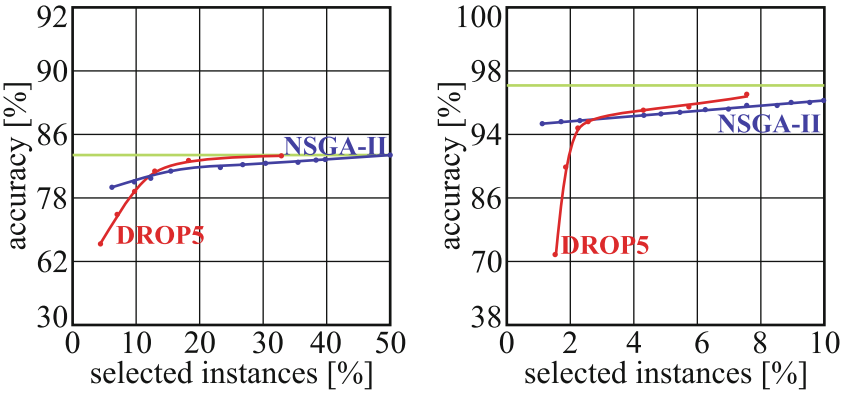

Fig. 5. The Pareto fronts for Magic (left) and Pageblocks (right) datasets. (Color figure online)

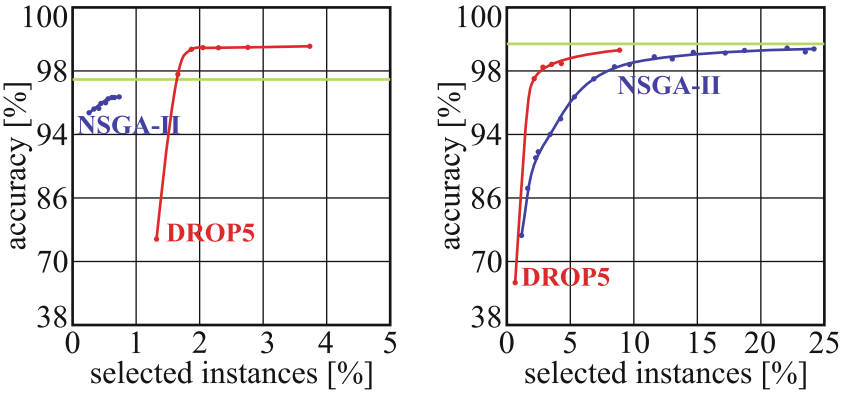

Fig. 6. The Pareto fronts for Twonorm (left) and Penbased (right) datasets. (Color figure online)

of DROP5 algorithms and multi-objective evolutionary optimization based on the NSGA-II algorithm. DROP5 was selected as it was one of the best non-evolutionary instance selection algorithms. The NGSA-II was chosen as one of the best multi-objective evolutionary algorithms. However, we had to adjust it to instance selection problem. The experiments showed the superiority of the evolutionary approach in terms of computational time and also terms of Pareto front position with the exception of the datasets with very high compression (about 1% remaining instances) obtained for high accuracy (above 98%), where the DROP5 ensembles were better. It is the problem with the NSGA-II method that in order to create an equally distributed front, it does not prefer the extreme solutions, so the front gets narrowed from both sides. We are currently working on this problem and we hope it will be solved soon (Fig. 3).

Acknowledgments. This work was supported by the NCN (Polish National Science Center) grant "Evolutionary Methods in Data Selection" No. 2017/01/X/ST6/00202.

References

1. Kordos, M.: Data selection for neural networks. Schedae Informaticae **25**, 153–164 (2017)
2. Arnaiz-González, Á., Blachnik, M., Kordos, M., García-Osorio, C.: Fusion of instance selection methods in regression tasks. Inf. Fusion **30**, 69–79 (2016)
3. Blachnik, M.: Ensembles of instance selection methods based on feature subset. IEEE Proc. Comput. Sci. **35**, 388–396 (2014)
4. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Hoboken (2001)
5. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley, Hoboken (2004)
6. Tomek, I.: An experiment with the edited nearest-neighbor rule. IEEE Trans. Syst. Man Cybern. **6**, 448–452 (1976)
7. Sebban, M., et al.: Stopping criterion for boosting based data reduction techniques: From binary to multiclass problem. J. Mach. Learn. Res. **3**, 863–885 (2002)
8. Garcia-Pedrajas, N.: Constructing ensembles of classifiers by means of weighted instance selection. IEEE Trans. Neural Netw. **20**, 258–277 (2009)
9. Blachnik, M., Kordos, M.: Bagging of instance selection algorithms. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2014. LNCS (LNAI), vol. 8468, pp. 40–51. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07176-3_4
10. García-Pedrajas, N., De Haro-García, A.: Boosting instance selection algorithms. Knowl.-Based Syst. **67**, 342–360 (2014)
11. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Mach. Learn. **38**, 257–286 (2000)
12. Olvera-López, A., Carrasco-Ochoa, J., Martínez-Trinidad, F., Kittler, J.: A review of instance selection methods. Artif. Intell. Rev. **34**(2), 133–143 (2010)
13. Garcia, S., Derrac, J., Cano, J.R., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 417–435 (2012)
14. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Boston (1989)
15. Lobo, F.G., Lima, C.F., Michalewicz, Z.: Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence, vol. 54. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-69432-8
16. Konak, A., Coit, D., Smith, A.: Multi-objective optimization using genetic algorithms: A tutorial. Reliab. Eng. Syst. Safety **91**, 992–1007 (2006)
17. Antonelli, M., Ducange, P., Marcelloni, F.: Genetic training instance selection in multiobjective evolutionary fuzzy systems: A coevolutionary approach. IEEE Trans. Fuzzy Syst. **20**(2), 276–290 (2012)
18. Tsaia, C.-F., Eberleb, W., Chu, C.-Y.: Genetic algorithms in feature and instance selection. Knowl.-Based Syst. **39**, 240–247 (2013)
19. Cano, J.R., Herrera, F., Lozano, M.: Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study. IEEE Trans. Evol. Comput. **7**(6), 561–575 (2003)

20. Cano, J.R., Herrera, F., Lozano, M.: Instance selection using evolutionary algorithms: an experimental study. In: Pal, N.R., Jain, L. (eds.) Advanced Information and Knowledge Processing, pp. 127–152. Springer, London (2004). https://doi.org/10.1007/1-84628-183-0_5
21. Derrac, J., et al.: Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection. Inf. Sci. **186**, 73–92 (2012)
22. Kordos, M.: Optimization of evolutionary instance selection. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2017. LNCS (LNAI), vol. 10245, pp. 359–369. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59063-9_32
23. Łapa, K., Cpałka, K., Hayashi, Y.: Hybrid initialization in the process of evolutionary learning. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2017. LNCS (LNAI), vol. 10245, pp. 380–393. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59063-9_34
24. Horoba, C., Numann, F.: Benefits and drawbacks for the use of e-dominance in evolutionary multi-objective optimization. In: Genetic and Evolutionary Computation Conference. ACM Press, pp. 641–680 (2008)
25. Alcala-Fdez, J., et al.: KEEL Data-Mining Software Tool and Data Set Repository http://sci2s.ugr.es/keel/datasets.php (2017)
26. Arnaiz-González, Á., Díez-Pastor, J.F., Rodríguez, J.J., García-Osorio, C.: Instance selection for regression: Adapting DROP. Neurocomputing **201**, 66–81 (2016)
27. Kordos, M., Blachnik, M., Perzyk, M., Kozłowski, J., Bystrzycki, O., Gródek, M., Byrdziak, A., Motyka, Z.: A hybrid system with regression trees in steel-making process. In: Corchado, E., Kurzyński, M., Woźniak, M. (eds.) HAIS 2011. LNCS (LNAI), vol. 6678, pp. 222–230. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21219-2_29
28. Kordos, M., Duch, W.: Variable step search algorithm for MLP training. In: The 8th IASTED International Conference on Artificial Intelligence and Soft Computing, Marbella, pp. 215–220, September 2004