Schedae Informaticae Vol. 25 (2016): 153–164
doi: 10.4467/20838476SI.16.012.6193

tfml 2017
theoretical foundations
of machine learning, Kraków

Data Selection for Neural Networks

Miroslaw Kordos
Department of Computer Science and Automatics
University of Bielsko-Biala
Willowa 2, 43-309 Bielsko-Biala
e-mail: *mkordos@ath.bielsko.pl*

Abstract. Several approaches to joined feature and instance selection in neural network leaning are discussed and experimentally evaluated in respect to classification accuracy and dataset compression, considering also their computational complexity. These include various versions of feature and instance selection prior to the network learning, the selection embedded in the neural network and hybrid approaches, including solutions developed by us. The advantages and disadvantages of each approach are discussed and some possible improvements are proposed.

Keywords: Neural Networks, Data Selection, Feature Selection, Instance Selection

1. Introduction

There are three main purposes of data selection: limiting the dataset size and thus accelerating the model learning process, removing noise from the data and thus improving the model predictive capabilities and making the data interpretation easier by humans [1, 2]. In this paper we discuss how data selection can be addressed in neural network learning. Since datasets consists of instances and the instances consist of features, the dataset size can be reduced by feature selection, instance selection or both. Moreover, the selections can be performed as well prior to neural network

Received: 11 December 2016 / Accepted: 30 December 2016

learning as by the neural network itself during the learning process. Our purpose is to show some interesting properties of data selection obtained with different feature or different instance selection methods, propose some improvements and discuss how to choose the optimal method.

Typically in data selection we fist obtain a little improvement of the prediction accuracy as we remove some data and then the accuracy slightly but constantly drops as more data is removed. Then after exceeding some limit the prediction accuracy begins constantly dropping and if we want to go further with compression we must choose some point depending on our preferences on the compression-accuracy plot.

Feature selection prior to model learning can be done either with filters or with wrappers. Wrappers really wrap the learning algorithm so it is not strictly done before the learning process but rather before deciding which will be the optimal configuration of the final model used for prediction. For the experiments we carefully chose some filter methods (section 2.).

Instance selection is discussed in this paper is more details than feature selection and we examine some possible improvements and parameterization of instance selection methods (section 3.).

We tried to answer the question: what is better feature selection or instance selection or both, and if both then how feature and instance selection influence each other and how they should be applied together for the best results (section 4.).

Another approach is not to perform any preliminary data selection but let the neural network select the relevant data itself. That includes selecting features (section 5.), instances (section 6.) and both of them (section 7.). That of course required us to make some adjustments of the neural learning process and error function.

Finally we experimentally compare the discussed approaches and their results (section 8.) and present the conclusions from this study (section 9.).

2. Feature Selection Before Network Learning

A detailed discussion of feature filters and wrappers can be easily found in literature [3, 4]. When the expert knowledge is available it can be used to make some preliminary feature selection [5, 6]. To select the methods we are going to use, we first performed some preliminary experiments with different feature selection and different instance selection methods using the RapidMiner software [7] and then we chose the methods that were among the best in term of the balance between compression, classification accuracy and running time.

W tested the following feature filters available in RapidMiner: Information Gain, Information Gain Ratio, Deviation, Chi Squared, Gini Index, Uncertainty, SVM, PCA, Correlation, Rule, Relief. We also tested three wrappers: forward selection, backward selection and evolutionary selection. Although the backward selection wrapper and evolutionary selection were able to discover more informative feature subsets, resulting in bit higher classification accuracy with the same number of features, their execution time was between two and four orders of magnitude longer, what in the case of the biggest data sets was totally impractical for our purposes. The results of the filter evaluation are presented in Table 1 in terms of the average classification

accuracy over the 10 datasets obtained in 10-fold crossvalidation and the average relative calculation time. This was done for the number of features being the nearest integer to 60% and 30% of the original number of features. Based on the test results, the SVM-based filter produced the best accuracy, but for the further experiments we chose the second filter in the accuracy ranking: Information Gain, because the SVM-based filter was over 100 times slower.

The information gain criterion IG is defined as the difference between the entropy before and after the optimal data split on feature f:

$$IG_f = -\sum_{i=1}^{C} p(c_i) \cdot log(p(c_i)) + \sum_{n=1}^{N} [\frac{v_n}{v} \sum_{i=1}^{C} p(c_{ni}) \cdot log(p(c_{ni}))] \quad (1)$$

where $p(c_i)$ is the probability that an instance belongs to class i and $p(c_{ni})$ is the probability that an instance within the bin n belongs to class i, v is the number of all instances and v_n is the number of instances in bin n, C is the number of classes and N is the number of bins. We did not use any methods that create new features (as PCA), because that makes the data interpretation and logical rule extraction in most cases totally impractical; the complexity of the obtained rules exceeds human capabilities of making any logical conclusions, what is important in most of our practical applications.

Table 1. Average values over the 10 datasets of classification accuracy of neural networks for the nearest integer of 60% and 30% of features (F60-acc, F30-acc) and execution time relative to Information Gain time with different feature filter methods using the RapidMiner implementation.

Method	F60-acc	F30-acc	time
no selection	92.74	92.74	0.0
Information Gain	92.12	91.02	1.0
Information Gain Ratio	92.37	89.80	1.0
Deviation	91.78	88.37	0.2
Chi Squared	91.82	90.48	0.8
Gini Index	92.07	89.52	1.1
Uncertainty	91.82	91.04	1.9
SVM	93.01	91.24	102
PCA	92.51	89.13	0.5
Correlation	89.35	87.40	0.1
Relief	93.02	88.27	245
Rule	92.15	88.44	16

3. Instance Selection Before Network Learning

Instance selection methods fall into two categories: noise filters and condensation methods. Noise filters remove noisy instances and thus improve the classification accuracy, but they compress the data very little. Condensation methods compress

the data and speed up the neural network (or any other classifier) training effectively but usually also decrease classification accuracy [8]. Frequently stronger compression causes stronger accuracy loss. If methods from both families are used then noise filters should be used first.

In the experiments, we first evaluated the performance of the instance selection algorithms implemented in the Information Selection extension for RapidMiner to decide, which one to use. Again our criteria were similar as with feature selection: the method should produce high compression, low accuracy loss and have short running time.

The description of many instance selection algorithms can be found in [9, 10]. The experimental comparison was also done in [11]. However, the authors used only small datasets, and thus, as the general tendency is preserved between their and our study, we obtained different results for our much larger datasets.

Also some methods of evolutionary-based instance selection were proposed in [12, 13, 14]. However, the computational time was several order of magnitude longer and for this reason we did not take them into account, although the authors reported their method more accurate. Another shortage of the evolutionary-based methods is that the selection is performed without letting us understand while particular instances are selected.

We evaluated the following instance selection algorithms using the RapidMiner Information Selection Extension: GE, RNG, CNN, IB2, IB3, DROP-3, RHMC, MC, ENN (which is the only noise filter on that list) and ENN. In the case of feature filters the results were very similar while averaged over the 10 datasets and thus selecting the proper feature filter was not so crucial. However, in the case of instance selection algorithms the differences are really very significant and here the proper choice is much more important.

Two of the best performing instance selection algorithms were ENN (Edited Nearest Neighbor) followed by IB3 [**?**] and DROP-3 [15]. However, they did not always provided the best accuracy (in some points ENN followed by ENN or even GE was better) and therefore we decided to use ENN with IB3 and ENN with a modified CNN (Condensed Nearest Neighbor).

The IB3 works in a similar ways as CNN. It takes the instances misclassified by k-NN and then it removes from the selected set the instances, which can be removed without the loss of classification accuracy.

DROP-3 [15] is a decremental algorithm, which first implements ENN and then examines which instances can be removed while their neighbors are still correctly classified without them. The examination is not in random order but staring from the instances situated furthest from other class instances (so called "enemies").

The compression of ENN with IB3 and DROP-3 was two times stronger than that obtained with ENN followed by CNN, but the accuracy was comparable. DROP-3 was situated almost on the same point on the compression-accuracy plot as ENN followed by IB3. Finally we decided to use first IB3 and then ENN with CNN. ENN removes noise by removing the instances which have a different class than predicted by the k-NN algorithm. Then CNN or IB3 removes the instances that can be eliminated without adversely affecting classification.

One of the basic problems with the condensation algorithms such as CNN, IB3 or the DROP family, is that there is usually no direct way to control how aggressively

they perform the instance elimination, unlike in feature filters, where we can select the desire numbers of remaining features. Thus we used two approaches to overcome the limitations: first bagging of instance selection methods [16] and in this work modification of the rejection criterion in the algorithm itself.

Table 2. Average values over the 10 datasets of classification accuracy of neural network, number of selected instances and execution time relative to CNN time of instance selection process for different instance methods using the RapidMiner implementation.

Method	accuracy	%Instances	time
no selection	92.74	100	-
GE	90.71	45	130
RNG	86.81	12	10
CNN	86.74	8.0	1.0
IB2	85.12	7.7	0.2
IB3	86.56	4.0	3.5
DROP-3	87.13	4.0	14**
MC	82-86*	3.5-20	8.0
RHMC	82-86*	3.5-20	8.1
ENN	93.17	90	1.0
ENN+CNN	87.44	7.1	2.0
ENN+IB3	87.15	3.9	4.5
ENN	93.17	90	1.0

* - MC and RHMC provide various results depending on adjustable parameters, but for the same number of instances they always provided lower accuracy then the four best algorithms.

** - DROP-3 was implemented as a Weka plugin, so the time comparison may not be adequate in this case.

In the case of bagging we use the same idea as bagging in classification, but instead of classifiers, instance selection algorithms constitute the ensemble. Then we establish a threshold of how many instance selection algorithms from the ensemble want to select a given instance [16]. Say, we had 10 algorithms. We decide that if $m = 5$ of them want to remove a given instance then the instance will get finally removed. But we can also use any arbitrary number for m between 1 and 10. The higher m the less aggressive is the selection - the compression will be weaker but the accuracy will be higher (which corresponds to a higher number of attributes kept in feature selection).

In our tests bagging worked very well, but its drawback was higher computational cost. Thus the other solution is to use variable m in the inner k-NN algorithms within CNN (and ENN - although it is not so crucial). For example, we can use the number of nearest neighbors $k = 9$. In the standard CNN an instance will get removed if it has the same class as more then $k/2$ of its neighbors, that is $m = 5$ in that case. If we increase m say to 8 then the instance will get removed if it has the same class as more then $m = 8$ of its neighbors, so the selection will be less aggressive and some of the instance situated close to class boundaries that with $m = 5$ would get removed will be

kept [17]. Again the compression will be weaker but the accuracy will be higher. This is shown in the pseudocode, where kNN(.) is the class of at least m of k neighbors in the k-NN algorithm and $C(\mathbf{x}_i)$ is the real class of instance $\mathbf{x}_i$. In particular k and m can be different for ENN and CNN.

Algorithm 1 The modified ENN+CNN algorithm

```
Input: original set T
Output: reduced set S (now empty)
for i = 1 ... numInstances do
  if C(x_i) ≠kNN(k, m, (T without x_i), x_i) then
    mark x_i for removal
  end if
end for
remove all marked instances from T
add randomly one instance from T to S
for i = 1 ... numInstances in T do
  if C(x_j) ≠kNN(min(k, sizeOf(S)), min(m, sizeOf(S)), S, x_i) then
    add x_i to S
  end if
end for
return S
```

4. Joined Feature and Instance Selection Before Network Learning

Our experiments with data selection as the preprocessing step showed that feature selection (FS) should be performed prior to instance selection (IS). We tested many different configurations, such as FS-IS, IS-FS, FS1-IS1-FS2-IS2, FS1-IS-FS2 and others. Also in the case of repetitive feature and instance selection we tried making the reduction stronger at each iteration. However, comparing that to the simplest strategy FS-IS, there was no significant difference on the compression-accuracy plot. Our explanation of that phenomenon is that in most data there is a higher percentage of irrelevant features than of noisy instances (most of the instances were removed because of their redundancy and not because of noise). Thus, we can perform efficiently feature selection using all instances, but less efficiently instance selection using all features. Moreover, several feature filters are based on some measure of correlation or some variants of information gain. Removing too many instances can make them work less efficiently. On the other hand most instance selection methods are based on the distance between the instances. If there are irrelevant features, we get not the optimal distance measures. A partial solution to that problem is multiplying the distance component in each feature direction by this feature importance obtained from some feature ranking.

Some papers proposed evolutionary optimization of feature and instance selection [18, 14], but as already mentioned in section 2., we did not consider this option

first because of the computational complexity and second because evolutionary optimization, similarly as feature construction methods (as PCA) does not enable us to understanding why particular results were obtained.

5. Feature Selection Embedded into Network Learning

Feature selection with neural networks can be done in several ways. The two basic approaches are by the analysis of weights, including pruning methods and by input data perturbances [19]. In perturbance analysis we replace the values of particular feature with random values in the test vectors and see how this influences the network accuracy. In weight analysis we assume that the less important features will generate smaller absolute values of weights and we can reject the features with the lower weighted sum of weights r_i (Eq. 2.). The weights can be also enforced to small values with a regularization term. The weight analysis was used in our experiments. A more complex method also consider the derivatives or the output neuron weights. Due to the non-linear transfer functions the results depend on the actual position on the transfer function and in classification task at the end of the training the position is predominantly in the saturated area, so there must be some more effort put into constructing an efficient solution. We used the following feature ranking measure:

$$r_i = \sum_{n=1}^{N} \frac{|w_{in}|}{\sum_{f=1}^{F} |w_{fn}|} \tag{2}$$

where r_i is the predictive power of the i-th feature, N is the number of hidden layer neurons, F is the numbers of features, w_{in} is the weight connecting the n-th hidden neuron with the i-th feature and w_{fn} $(= 0...F)$ is the f-th weight of the n-th hidden neuron.

The random perturbances of single feature values were not evaluated experimentally, because in the experiments we were removing features (thus setting them to zeros) before network training and that already partially corresponds to the perturbance analysis.

6. Instance Selection Embedded into Network Learning

We use the MLP neural network with hyperbolic tangent transfer function and with the number of output neurons equal to the number of classes. Most of the existing neural network training algorithms can be used. The error for a single vector is given by the following formula:

$$Error(x_v) = \sum_{c=1}^{C} (y_{ac} - y_{ec})^2 \tag{3}$$

where C is the number of classes, y_{ac} is the actual value of $c-th$ output neuron signal and y_{ec} is the expected value of $c-th$ output neuron signal (which is 1 if the

current instance class is represented by the $i-th$ output neuron and -1 otherwise). We assume that a vector is classified correctly if the neuron associated with its class gives a higher signal than any other output neuron.

If an instance is classified incorrectly, the error the network gives as a response to that instance is high. If an instance is classified correctly and is situated far from a class boundary, the error obtained for that instance is very low (due to the hyperbolic tangent transfer function shape). Thus we can remove the instances with the highest error (greater than $maxError$), as they are outliers and these with the lowest errors (lower that $minError$), as they do not help to determine the proper decision boundary [20]. By adjusting the two parameters $maxError$ and $minError$ we can regulate the compression level. That however cannot be done from the very beginning of the network training, because at that stage the network produces more or less random errors for each instance, as the learning starts with random weights. As the training progresses, we can gradually decrease $maxError$, starting from its maximal possible value ($4C$, assuming MSE error measure). $minError$ does not require gradual increasing but can be set at the end of network learning. $maxError$ improves mostly accuracy and compression very little [20]. We set $maxError = 1.6 + 0.16 \cdot numberOfClasses$. The influence of $minError$ values on the data compression and accuracy together with feature selection is experimentally evaluated in section 8.

7. Joined Feature and Instance Selection Embedded into Network Learning

To determine the optimal order of joined feature and instance selection with neural networks, we conducted experiments trying feature selection first, instance selection first and simultaneous feature and instance selection. As in the case of the selection prior to network learning, the results confirmed that the best option is to perform feature selection first and then instance selection. Thus the network training consists of three parts: 1. standard network training, 2. removal of irrelevant features, 3. removal of irrelevant instances. After the second step the training can either continue or it can be restarted from random weights. Better results were obtained with the restart.

We observed in the experiments that instance selection embedded in neural network worked well, but feature selection was in some cases as good as done with feature filters and in some cases less effective. For that reason we added another option for feature selection. First we build a simple neural network with three hidden units and train this network separately on each feature. We use early stopping, so that we can measure the classification accuracy on a training set, without the need of crossvalidation. Then we sort the features by the classification accuracy. Then we add the features to the reduced dataset starting from the most informative one. However, before we add the next one, we calculate its correlation with all the already added feature. If the correlation with at least one of them is higher than the threshold, we reject this feature. This appeared to be the most accurate option.

8. Experimental Evaluation

The purpose of the experiments was to evaluate particular methods in terms of classification accuracy and data compression and determine the Pareto-line for each method. The Pareto-line is a line in the compression-accuracy coordinates, shown in Fig. 1 that connects the points with the best compression for a given accuracy and best accuracy for a given compression, so that no other points exist that can improve both. The compression is defined as percentage of remaining features multiplied by percentage of remaining instances (lower value is better).

In the experiments we used RapidMiner for feature selection and some of the instance selection before the network training. All the other experiments, especially data selection embedded into neural network were performed in our own program. All the software we used can be downloaded from our web page at *www.kordos.com/tfml2017*.

We trained the networks with the R-prop algorithm. We used networks with one hidden layer. The numbers of neurons in the hidden layer was equal to the geometric mean of the number of inputs and number of classes. We performed the experiments on 10 classification datasets from the Keel Repository [21]: Ionosphere (351,33,2), Image Segmentation (210, 18, 7), Magic (19020, 20, 2), Thyroid (7200, 21, 3), Page-blocks (5472, 10, 5), Shuttle (57999, 9, 7), Sonar (208, 60, 2), Satellite Image (6435, 36, 6), Penbased (10992, 16, 10), Ring (7400, 20, 2). The numbers in the brackets are: number of instances, number of features, number of classes. All the experiments were performed in 10-fold crossvalidation and repeated 10 times. The purpose of repeating the experiments 10 times was to average over the initial random network weights and thus to ensure more stable and reliable results.

Table 3. Average values over the 10 datasets of classification accuracy of neural networks (F100-A, F60-A, F30-A) and number of selected instances (F100-I, F60-I, F30-I) for respectively 100%, 60% and 30% of features with three data selection methods (the real numbers of features were the nearest integers to these percentages).

Method	IS	F100-A	F100-I	F60-A	F60-I	F30-A	F30-I
FS: Inf. Gain	no selection	92.74	100	92.12	100	91.02	100
IS: ENN+CNN	m=8, k=9	93.01	35.22	92.30	29.11	91.03	27.81
with variable	m=7, k=9	92.65	18.91	91.58	16.15	88.76	15.11
m in k-NN	m=5, k=9	87.44	7.11	87.11	5.89	87.72	6.41
IS: ENN+IB3	ENN+IB3	87.15	3.85	86.88	3.98	87.01	4.95
FS in separate	no selection	92.74	100	92.90	100	91.45	100
network	minE=0.03	93.02	38.45	92.30	32.98	91.23	36.40
IS embedded	minE=0.1	92.64	19.98	91.91	22.12	88.76	26.78
into NN	minE=0.3	89.05	7.11	88.23	5.89	88.25	10.23
FS embedded	no selection	92.74	100	92.12	100	88.34	100
into NN	minE=0.03	93.02	8.45	92.15	33.15	88.94	38.14
IS embedded	minE=0.1	92.61	19.98	91.05	24.98	87.02	14.18
into NN	minE=0.3	89.05	7.11	87.91	9.15	86.15	10.78

The standard deviations were between 0.5 for the largest to 3.0 for the smallest datasets. The T-test confirmed that the differences in Table 3. are statistically

significant. (For example for the accuracies of 90.00 and 91.00 with 100 cases, the p-value of 0.05 is obtained with standard deviation of 3.585.)

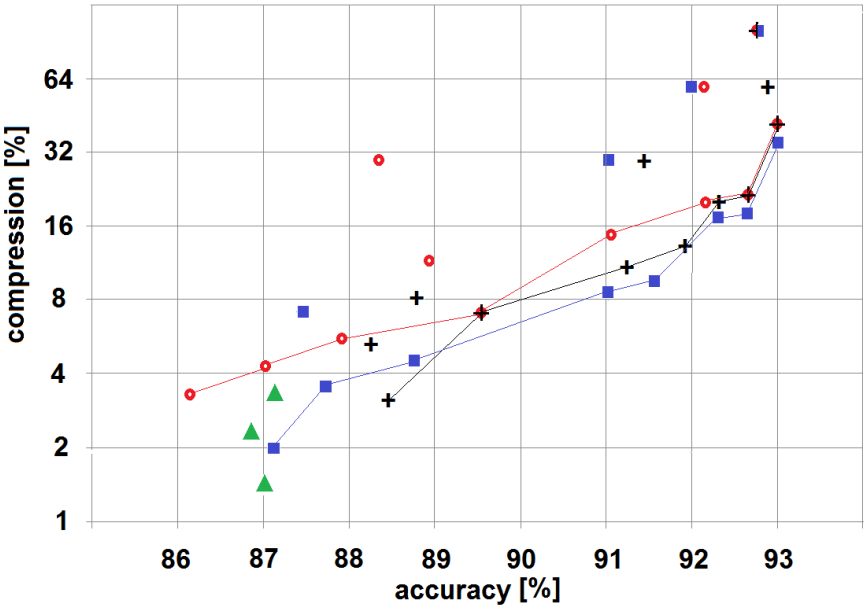

Figure 1. Compression (percentage of remaining features times percentage of remaining instances) vs classification accuracy. The compression axis is in logarithmic scale. A Pareto line is shown separately for each method (square = FS: Inf. Gain, IS: Mod. ENN+CNN, cross = FS in sep. network, IS embedded, circle = FS and IS embedded, triangle: FS: Inf. Gain, IS: ENN+IB3).

9. Conclusions

For the biggest data sets we were able to remove about 98-99% of instances without noticeably accuracy loss, but for smaller datasets the reduction was much weaker.

The most effective data selection is performed by feature selection followed by instance selection. This is true as well as for the selection prior to network training as embedded into the neural network. Currently the Pareto line for the selection with information gain and ENN+IB3 and then modified ENN+CNN is situated closest to the right lower corner, so this method looks the better. However, each of the methods have some strengths.

Feature ranking obtained by learning a simple neural network on a single features with removal of highly correlated features worked very well. The standard feature rankings, as information gain, were on the second place, while feature selection by neural network weight analysis on the third place. However, the last methods can be further enhanced by considering the data flow through the entire network, not only

the input to hidden weights and thus may produce better results, which will the topic of a further study.

Embedding noise reduction into the neural network learning process gives usually very good results. That can be attributed to the shape of decision boundaries, where the k-NN algorithm have the tendencies to smooth the edges.

Instance selection as noise removal works quite well in each case. There is however one problem with instance selection as data compression. Both DROP-3 and the instance selection based on the network error overcomes the shortage of CNN that it works in a random order, as they both preserve more of the instances situated close to class boundaries. However, both of the methods rely on the distance to the opposite class measured either directly (CNN, IB3 and the DROP family) or by the instance location reflected by error produced by the hyperbolic tangent function transformation. But both of the approached do not consider the fact, that the distance between opposite class instances maybe different in different areas of the input spaces and thus sometimes tend to remove rather the instances closest to the boundary, even if they are hidden between the "first row" of instances then the instances that are further, but in the first row and thus needed to preserve the boundary. That is considered by other instance selection methods, which examine the classes of neighbor instances of Voronoi cells, but in spite of that they do net perform better. Finding an effective solution to this problem is still open.

It is likely that the results can be further improved if the variable m in the k-NN algorithms is used also with IB3 and DROP-3 algorithms, which have better compression than CNN with comparable accuracy. That will be another topic of our future research.

10. References

[1] Kordos, M., Blachnik, M., Bialka, S., *Instance selection in logical rule extraction for regression problems.* Lecture Notes in Artificial Intelligence, 2013, **7895**, pp. 167–175.

[2] Blachnik, M., Kordos, M., *Simplifying SVM with weighted LVQ algorithm.* Lecture Notes in Computer Science, 2011, **6936**, pp. 212–219.

[3] Liu, H., *Computational Methods of Feature Selection.* Chapman and Hall, 2007.

[4] Stanczyk, U., Jain, L.C., *Feature Selection for Data and Pattern Recognition.* Springer, 2015.

[5] Uribe, C., Isaza, C., *Expert knowledge-guided feature selection for data-based industrial process monitoring.* Rev. Fac. Ing. Univ. Antioquia, 2012, **65**, pp. 112–125.

[6] Kordos, M., Cwiok, A., *A new approach to neural network based stock trading strategy.* Lecture Notes in Computer Science, 2011, **6936**, pp. 429–436.

[7] Hofmann, M., Klinkenberg, R., *RapidMiner: Data Mining Use Cases and Business Analytics Applications.* Chapman and Hall/CRC, 2016.

[8] Sun, X., Chan, P.K., *An analysis of instance selection for neural networks to improve training speed.* International Conference on Machine Learning and Applications, 2014, pp. 288–293.

[9] Garcia, S., Derrac, J., Cano, J.R., Herrera, F., *Prototype selection for nearest neighbor classification: Taxonomy and empirical study.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, **34**, pp. 417–435.

[10] Olvera-Lapez, J.A., Carrasco-Ochoa, J.A., Martin, J.F., Kittler, J., *A review of instance selection methods.* Artificial Intelligence Review, 2010, **34**, pp. 133–143.

[11] Grochowski, M., Jankowski, N., *Comparison of instance selection algorithms.* Lecture Notes in Computer Science, 2004, **3070**, pp. 580–585.

[12] Antonelli, M., Ducange, P., Marcelloni, F., *Genetic training instance selection in multiobjective evolutionary fuzzy systems: A coevolutionary approach.* IEEE Transactions on Fuzzy Systems, 2012, **20**, pp. 276–290.

[13] Anwar, I.M., Salama, K.M., Abdelbar, A.F., *Instance selection with ant colony optimization.* Procedia Computer Science, 2015, **53**, pp. 248–256.

[14] Derrac, J., Cornelis, C., Garcia, S., Herrera, F., *Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection.* Information Sciences, 2012, **186**(73–92).

[15] Wilson, D.R., Martinez, T.R., *Reduction techniques for instance-based learning algorithms.* Machine Learning, 2000, **38**, pp. 257–286.

[16] Blachnik, M., Kordos, M., *Bagging of instance selection algorithms.* Lecture Notes in Computer Science, 2014, **8468**, pp. 40–51.

[17] Kordos, M., *Instance selection optimization for neural network training.* Lecture Notes in Artificial Intelligence, 2016, **9692**, pp. 610–620.

[18] Tsaia, C.F., Eberleb, W., Chu, C.Y., *Genetic algorithms in feature and instance selection.* Knowledge-Based Systems, 2013, **39**, pp. 240–247.

[19] Leray, P., Gallinari, P., *Feature selection with neural networks.* Behaviormetrika, 1999, **26**, pp. 145–166.

[20] Rusiecki, A., Kordos, M., Kaminski, T., Gren, K., *Training neural networks on noisy data.* Lecture Notes in Artificial Intelligence, 2014, **8467**, pp. 131–142.

[21] Alcala-Fdez, J., et al., *Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework.* http://sci2s.ugr.es/keel/datasets.php, Journal of Multiple-Valued Logic and Soft Computing, 2011, **17**, pp. 255–287.