

# Instance Selection Optimization for Neural Network Training

Mirosław Kordos

University of Bielsko-Biala, Department of Computer Science and Engineering,  
Bielsko-Biala, Willowa 2, Poland: mkordos@ath.bielsko.pl

**Abstract.** Performing instance selection prior to the classifier training is always beneficial in terms of computational complexity reduction of the classifier training and sometimes also beneficial in terms of improving prediction accuracy. Removing the noisy instances improves the prediction accuracy and removing redundant and irrelevant instances does not negatively effect it. However, in practice the instance selection methods usually also remove some instances, which should not be removed from the training dataset, what results in decreasing the prediction accuracy. We discuss two methods to deal with the problem. The first method is the parameterization of instance selection algorithms, which allows to choose how aggressively the instances are removed and the second one is to embed the instance selection directly into the prediction model, which in our case is an MLP neural network.

## 1 Introduction

There are two purposes of instance selection: to decrease noise in the data and to reduce the data size. One of the most popular instance selection methods for noise filtering is ENN (Editted Nearest Neighbor) [21] and for data size reduction is CNN (Condensed Nearest Neighbor) [9]. Although ENN and CNN as single methods are quite simple and there exists methods that perform better both in terms of noise filtering and data compression, when they are applied sequentially ENN followed by CNN they produce exceptionally good results. First ENN removes noise, this is the instances that are wrongly classified by k-NN. Then CNN removes redundant instances in the following way: first it selects one instance and then if the next instance is correctly classified by k-NN using the already selected instances as the training set, it is considered redundant and not selected. If classified incorrectly it is added to the selected instances. Then the process repeats with each remaining instance. The ENN and CNN algorithms are well described in the literature [21],[9] and in [18], [17] and [12] the interesting reader can find a detailed comparative study of many instance selection methods.

The standard instance selection methods are rather filters than wrappers and thus have no real-time adjustment to the classifier performance. In practice there are two problems: some of the instances that should be selected do not get selected and some of the instances that should not be selected get selected. That is because ENN and CNN consider only the local neighborhood of each instance and do not take into account how the classification model works. What it means in practice is that there is no guarantee that the set of selected instances will be the optimal one for a given classifier.

Recently there have been several approaches to improve instance selection methods by addressing some of their shortages. In [2] Antoneli et al. presented a genetic approach using a multiobjective evolutionary algorithms for instance selection. In [8] Guillen et al. presented the use of mutual information for instance selection. Their proposition was based on a similar principles as the  $k$ -NN based instance selection, but after finding the nearest neighbors of a given example instead of using  $k$ -NN to predict its output value, the mutual information (MI) between that example and each of its neighbors was determined. In the next step the loss of mutual information with respect to the neighbors was calculated. If the loss was similar to the example neighbors then this example was selected to the final dataset. The method was experimentally evaluated on artificially generated data with one and two input features. Then in [19] the idea was extended to instance selection in time series prediction.

In [16] a class conditional instance selection (CCIS) was presented. CCIS creates two graphs: one for the nearest neighbors of the same class and one for other class instances than the current example. Next a scoring function, which is based on the distances in graphs, is used to determine the selected instances. In [3] the authors proposed to use ant colony optimization with one classification model used in the instance selection process and another model as the final predictor. In [1] the authors analyzed how instance and feature selection influence neural network performance. In [4] a method to reduce the number of support vectors in SVM training was considered.

From our perspective, very interesting ideas were presented in [20]. The authors designed an instance selection method that took into account the decision boundaries of neural networks like distance from decision boundary, dense regions and class distributions and they proposed an instance selection method adjusted to neural network properties. Their method consisted of two parts: removing far instances and removing dense instances. In the far part they calculated the distance between each instance and the closest instance of the opposite class (the far distance) and removed the instances for which the distance was farther then the average + standard deviation of the far distances. In the dense part they calculated the distance between each instance and the closest instance of the same class (the dense distance) and iteratively removed the instances for which the distance was closer then the average of the dense distances, starting from those with the smallest dense distance and updating the average dense distance at every iteration.

We discuss two methods to deal with the problem. The first method is the parameterization of instance selection algorithms, which allows to choose how aggressively they remove the instances and the second one is to embed the instance selection directly into the prediction model, which in our case is an MLP neural network. To use the MLP neural network for an on-line instance selection no adjustments are required to the error function, neuron transfer functions, to the learning algorithm or to the network architecture.

## **2 Instance Selection Prior To Neural Network Training**

The common problem with instance selection performed before classification is reduction of the classification accuracy. That is true not only for ENN and CNN but also

for many other instance selection methods. That may be not of a great concern if the purpose is primary to reduce the dataset size and the accuracy reduction is very little. However, frequently the objective is to obtain high accuracy at the first place and then the possible data size reduction.

As the instance selection method we use a modified ENN (Editted Nearest Neighbor) followed by a modified CNN [9]. The choice of the instance selection algorithms is dictated by their simplicity and by the fact that when they are both applied in this order, they perform exceptionally well. First ENN removes noise, then CNN removes redundant instances. k-NN can be used as the final prediction algorithm and as the internal CNN and ENN algorithm. In both cases it works best with an optimal  $k$  value, which according to our experiments for a broad range of problems can be set to  $k=9$  and moreover this algorithm is not very sensitive to little changes of  $k$ . For that reason in all the experiments we will keep  $k=9$  (at the beginning of CNN, when the number of selected instances is fewer than 9 we use all the already selected instances as the neighbors).

However, to address the above mentioned problem we must make some modifications to ENN: we will require that in order for an instance to be rejected by ENN, it must have different class than more as  $m=5$  of its neighbors, which will cause only more outlying instances to be removed. If we want to reject even more outlying instances, we can rise the requirement to  $m=7$  neighbors of a different class or even 8 or all 9 (which will be the weakest instance selection, preserving most instances in the selected set).

In a similar way, we can apply modifications to CNN to require a different numbers of the instance neighbors to belong to the same class to reject the instance. The more instances of the same class are required the weaker the selection is, resulting in rejecting only the instances situated far from the decision boundaries (and thus very few or none of their neighbors belong to a different class). That is exactly why we need to perform noise reduction first. If some noisy instances, this is instances of a different class surrounded by the current class instances, are still present, they would not allow us to successfully remove the redundant data points, especially when we rise the requirements for the number of the same class neighbors.

All possible values of  $m=5, 6, 7, 8$  and 9 for ENN and CNN lead to 25 different combinations. Since there is no place in the paper to show the results of all 25 combinations, we selected four of them in such a way that  $m$  for ENN ( $m_{enn}$ ) equals  $m$  for CNN ( $m_{cnn}$ ) and both  $m$  increase from 5 to 8. We found these four combinations to be most representative, well reflecting the general trend and the easiest to interpret.

The modified ENN+CNN algorithm pseudo-code is presented in Algorithm 1, where  $\mathbf{T}$  is the original training set,  $\mathbf{P}$  is an intermediary set, which is an output in ENN and input to CNN and  $\mathbf{S}$  is the selected training set.  $\bar{C}(x_i)$  is the class of a at least  $m_{ENN}$  or  $m_{CNN}$  nearest neighbors of the instance  $x_i$  reduced to two-class problem: the same class vs. different class,  $C(x_i)$  is the class of the instance  $x_i$ ,  $k$  is the  $k$  in k-NN.  $\mathbf{S} \leftarrow \mathbf{S} \cup x_i$  means that the vector  $x_i$  is added to the set  $\mathbf{S}$  and  $\mathbf{P} \leftarrow \mathbf{P} \setminus x_i$  means that vector  $x_i$  is removed from the set  $\mathbf{P}$ .  $k_0$  and  $m_0$  are used to set  $k$  and  $m_{CNN}$  to smaller values when the number of already selected instanced in the new training set is less than 9.

---

**Algorithm 1** The modified ENN+CNN algorithm

---

**Require:**  $\mathbf{T}, k, m_{ENN}, m_{CNN}$ 

```
 $n \leftarrow |\mathbf{T}|;$   
 $\mathbf{P} \leftarrow \mathbf{T};$   
for  $i = 1 \dots n$  do  
   $\bar{C}(\mathbf{x}_i) = \text{kNN}(k, m_{ENN}, (\mathbf{T} \setminus \mathbf{x}_i), \mathbf{x}_i);$   
  if  $C(\mathbf{x}_i) \neq \bar{C}(\mathbf{x}_i)$  then  
     $\mathbf{P} \leftarrow \mathbf{P} \setminus \mathbf{x}_i$   
  end if  
end for  
 $\mathbf{S} \leftarrow \text{empty};$   
 $\mathbf{S} \leftarrow \mathbf{S} \cup \mathbf{x}_1;$   
 $flag \leftarrow \text{true}$   
while  $flag$  do  
   $flag \leftarrow \text{false}$   
   $n \leftarrow |\mathbf{P}|$   
  for  $i = 1 \dots n$  do  
     $k_0 = \min(k, \text{sizeof}(S))$   
     $m_0 = m_{CNN} \cdot k_0 / k$   
     $\bar{C}(\mathbf{x}_i) = \text{kNN}(k_0, m_0, \mathbf{S}, \mathbf{x}_i)$   
    if  $\bar{C}(\mathbf{x}_i) \neq C(\mathbf{x}_i)$  then  
       $\mathbf{S} \leftarrow \mathbf{S} \cup \mathbf{x}_i;$   
       $\mathbf{P} \leftarrow \mathbf{P} \setminus \mathbf{x}_i$   
       $flag \leftarrow \text{true}$   
    end if  
  end for  
end while  
return  $\mathbf{S}$ 
```

---

### 3 Instance Selection Embedded in Neural Network Training

It seems reasonable to embed instance selection in the classifier learning process. One advantage is solving the problem of different decision borders of k-NN and a neural network. Another advantage is that in some cases the instance can be not removed totally, but be treated differently in the model learning - as the less liable example and thus less contributing to the model parameters. Still another advantage is the possibility of assessing during the network training how the selection influences the results and adjust the selection accordingly. The drawback of this approach may be in some cases higher computational complexity of the classification process than the joint complexity of the prior instance selection followed by learning the classifier on the reduced set. This is especially evident for large datasets, where k-NN can be efficiently accelerated by methods like k-means clustering and then searching for the nearest neighbors only within one cluster, KD-Tree [5] or Local Sensitive Hashing [11].

We use an MLP neural network with hyperbolic tangent transfer function and we have as many neurons in the output layer as the number of classes. Most of the existing

neural network training algorithms can be used. When a vector is given to the trained network inputs, the output neuron associated with this vector class gives signal = 1 and all the other output vectors associated with different classes give the output signal = -1. The error for a single vector  $\mathbf{x}_i$  used for instance selection is given by the following formula:

$$Error(\mathbf{x}_i) = \sum_{i=1}^n (y_{ai} - y_{ei})^2 \quad (1)$$

Where  $n$  is the number of classes, which equals the number of output layer neurons,  $y_{ai}$  is the actual value of  $i$ -th output neuron signal and  $y_{ei}$  is the expected value of  $i$ -th output neuron signal (which is 1 if the current instance class is represented by the  $i$ -th output neuron and 0 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 of the training set is classified incorrectly by a trained neural network, the error that the network gives as a response to that instance is high ( $Error(\mathbf{x}_i) > maxError$ ). On the other hand if an instance is classified correctly and is situated far from a classification boundary (because of the hyperbolic tangent transfer function), the network error obtained for that instance will be very low ( $Error(\mathbf{x}_i) < minError$ ). Thus a solution is to remove from training set  $\mathbf{T}$  the instances that produce very high and very low errors. The pseudo-code for the embedded instance selection is shown in Algorithm 2.

---

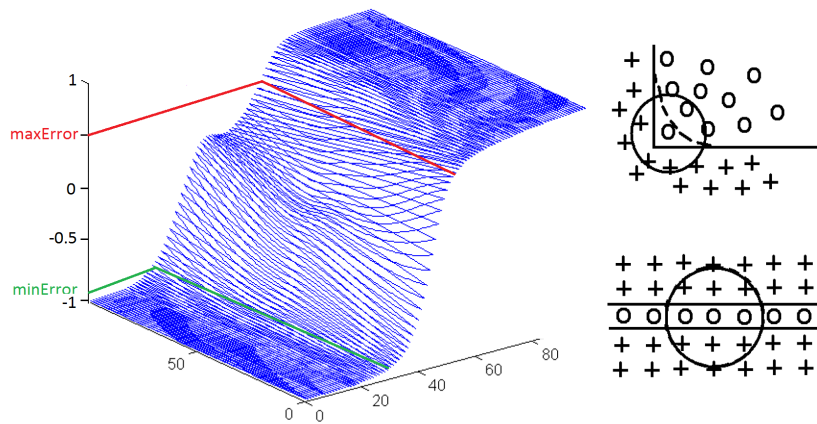
**Algorithm 2** Instance selection embedded in neural network training

---

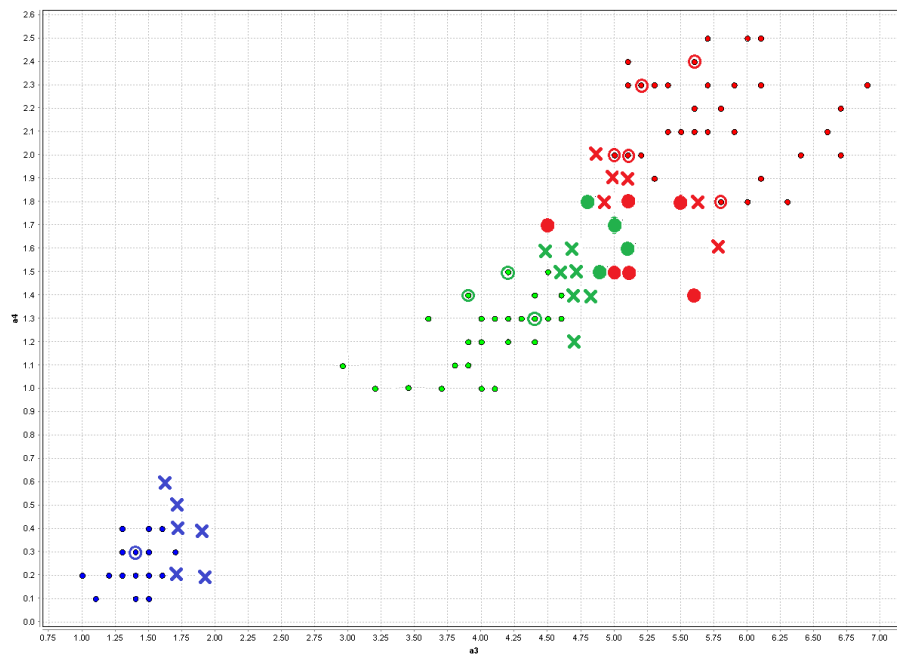
**Require:**  $\mathbf{T}, minError, maxError$   
 $n \leftarrow |\mathbf{T}|$ ;  
train the network on  $\mathbf{T}$   
**for**  $i = 1 \dots n$  **do**  
    **if**  $Error(\mathbf{x}_i) > maxError$  or  $Error(\mathbf{x}_i) < minError$  **then**  
         $\mathbf{T} \leftarrow \mathbf{T} \setminus \mathbf{x}_i$   
    **end if**  
**end for**  
**if** restart = true **then**  
    retrain the network on  $\mathbf{T}$  from random weights  
**else**  
    continue training the network on  $\mathbf{T}$  from the current weights  
**end if**

---

There are several parameters in this instance selection. The first two parameters are the  $maxError$  and the  $minError$ . The next one is the time point in the network training, when the errors should be calculated and the instances rejected. And the final one is what to do after the instances get removed: train the network further from the current point or start the training anew. For  $maxError$  we use a value be greater than 1 and smaller than 2 due to the shape of hyperbolic tangent for two class problems, but for multiply classes it may be set to higher values.



**Fig. 1.** Left: MLP neural network decision border with *minError* and *MaxError* shown for the class for which an expected neuron signal is -1. Right: Examples of decision borders that make some instances to be incorrectly classified by k-NN.



**Fig. 2.** The instances of Iris dataset selected by CNN (dot inside circle), by neural network (cross) as those with low error values and by both (big solid circle). Neural network selected instances more correctly at the border of red and green class and of blue class, only failed to select border instances of the green class from the blue class side.

The rationale behind using the *maxError* and not simply rejecting all misclassified instances, is that the instances that are situated close to the decision boundaries may be frequently misclassified, although it is rather due to the neural network properties than due to the instances being noisy. However, if a single instance of one class is in the middle of different class instances then it is surely noise and the error value produced by the network for such an instance will be higher than for the misclassified border instances. However, the *minError* depends more on the neuron weights values and thus a better solution than using a constant value is to use a relative value in relation to the error the network makes on other examples. We use for *minError* some percentage of the average error values of all correctly classified instances of a given class. In the experiments we use four different values of *maxError* and *minError*

While removing irrelevant examples as those, on which the neural network makes the least error, the examples that get removed are those far from the decision border, so those that are not necessary to determine the decision border and thus the decision border remains intact. But while removing them with CNN we have no guarantee that only the irrelevant ones will get removed, because that may depend on the order of the instances being considered. Let us illustrate this in Fig. 2.

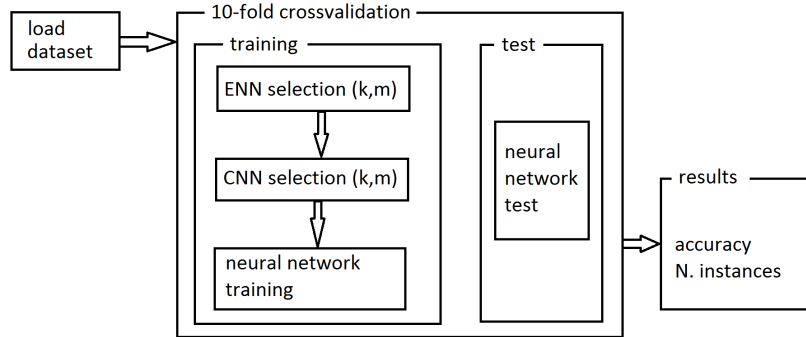
In the case of noise reduction also different examples may get removed by ENN and the neural-network embedded noise reduction, however in this case the difference is not so dramatic, but rather determined by the decision boundary shape of particular learning algorithms. In k-NN, which is the baseline algorithm for ENN mostly the edges get smoothed and narrow stripes get removed (see Fig. 1. right). Neural networks are able to accommodate more complex decision boundaries and particularly if the neuron weights have enough big values and the problems shown in Fig. 1. right can be eliminated, although on the other hand it may lead to over-fitting if there are many neurons and large weight values. To overcome this weight pruning methods can be used.

## 4 Experimental Evaluation

We conducted the experiments with several datasets from the UCI Machine Learning Repository [6]. The datasets were selected to cover different levels of noise, of leaning difficulties (the neural network could achieve the prediction accuracy from 0.6 to 1.0 depending on the dataset) and different number of classes. In this way the approach can be tested for various types of problems. The following dataset were used: Banknote Authentication (4 features, 1372 instances, 2 classes), Climate Simulation Crashes (4 features, 1372 instances, 2 classes), Image Segmentation (19 features, 2310 instances, 7 classes), Satellite Image (36 features, 6435 instances, 7 classes), Vehicle (18 features, 846 instances, 4 classes) and Yeast (8 features, 1484 instances, 10 classes).

We implemented the algorithms in C# language. As the network learning algorithms we use VSS [13], which uses a search-based approach for finding the optimal path downwards the error surface [14]. The software can be downloaded from [7]. Also some tests were performed in RapidMiner [10]. We run each test in five 10-fold cross-validations (50 runs together). Fig. 3 shows the diagram representing the experimental setup, where  $k=9$  the values  $m$  for ENN and CNN are the same and change from 5 to 8. The same setup is used for neural network based instance selection and in this case the

ENN and CNN blocks are not used. In this case the instance selection was performed at the final stage of the network learning. Then the network learning was either restarted (**res** in tables 1 and 2) from random weights or the network trained further "incrementally" (**inc** in tables 1 and 2) from the point where the selection was done - in both cases using the selected instances only.



**Fig. 3.** Processes used to evaluate and validate dataset assessment based on compression. Either ENN and CNN blocks are used or the instance selection embedded in a neural network training.

**Table 1.** Classification accuracy

method/dataset	Banknote	Climate	ImgSegm	SatImage	Vehicle	Yeast
no selection	99.5	95.1	96.4	90.6	81.6	<b>60.0</b>
ENN-CNN 5/9	97.5	94.0	92.3	86.5	81.7	55.7
ENN-CNN 6/9	98.2	94.3	94.6	87.8	<b>82.0</b>	55.9
ENN-CNN 7/9	<b>99.8</b>	95.1	95.7	90.2	78.4	56.2
ENN-CNN 8/9	98.2	94.5	<b>96.6</b>	89.8	80.3	59.0
NN-inc 0.02/1.7	99.4	94.0	94.8	89.3	76.1	59.3
NN-inc 0.05/1.6	99.8	92.2	87.3	91.0	77.2	55.2
NN-inc 0.10/1.5	-	90.8	81.6	91.6	78.1	-
NN-inc 0.20/1.4	-	87.0	70.3	90.2	73.4	-
NN-res 0.02/1.7	99.8	<b>96.2</b>	94.8	92.3	81.1	59.0
NN-res 0.05/1.6	99.8	96.0	94.3	<b>92.4</b>	80.6	56.5
NN-res 0.10/1.5	-	94.8	93.6	91.6	78.1	-
NN-res 0.20/1.4	-	93.4	80.1	91.3	81.4	-
ENN-CNN std. dev.	1.3	2.3	1.2	1.9	2.8	3.5
NN std. dev.	1.6	4.0	4.4	2.0	3.5	4.5

The standard deviations of accuracy were relatively constant for each dataset and instance selection parameters, so they are reported once in the bottom row of tables 1



**Table 2.** Ratio of selected instances to all instances

method/dataset	Banknote	Climate	ImgSegm	SatImage	Vehicle	Yeast
no selection	100	100	100	100	100	100
ENN-CNN 5/9	3.03	2.80	12.0	10.4	20.7	17.2
ENN-CNN 6/9	3.91	9.18	19.0	11.9	40.4	22.0
ENN-CNN 7/9	5.10	18.5	27.2	18.0	40.3	25.9
ENN-CNN 8/9	7.22	32.9	35.6	35.3	62.4	38.4
NN 0.02/1.7	22.1	11.6	14.8	88.2	77.1	33.3
NN 0.05/1.6	17.8	7.96	9.75	63.7	53.0	24.2
NN 0.10/1.5	-	5.64	7.17	41.7	42.6	-
NN 0.20/1.4	-	2.53	4.37	28.1	34.5	-
ENN-CNN std. dev.	1.6	1.5	0.6	1.8	0.9	4.0
NN std. dev.	1.6	0.6	0.6	1.9	1.5	4.5

and 2 as the average standard deviation of the 50 runs (five 10-fold crossvalidations) of each set of instance selection parameters. In ENN+CNN selection the first value is  $m$  (here  $m = m_{ENN} = m_{CNN}$ ) and the second  $k$ . For neural network based selection the first value is  $minError$  and the second is  $maxError$ .  $maxError$  is presented in absolute values and  $minError$  in the fraction of the average error value.

## 5 Conclusions

It was in several cases possible to obtain higher classification accuracy than without instance selection using about 20-30% of the training vectors and always to significantly reduce the dataset size with only a very little accuracy drop - much less than caused by ENN+CNN with standard parameters. However, other experiments showed that ENN with default parameters is very good at removing noise artificially added to the training dataset and thus improving results on such data. However, in the case of the datasets used here, ENN seems to perform too strong instance selection and in particular it removes too many instances situated close to decision boundaries, what shifts the decision boundaries causing decrease in the classifier performance. Increasing the required  $m$  we can get removed only the instances situated inside other class and thus only noise is removed and the decision borders are left intact. ENN selection does not have much impact on the data size reduction. In the experiments ENN usually reduced the data size about 5% and CNN up to 85%. Also standard CNN selection tends to shift the decision boundaries due to the removal of the instances situated close to the boundaries. And again increasing  $m$  from the default 5 to 7 or to 8 helped, but the total number of selected instances at least doubled. However, if decreasing the dataset size is as important as accuracy improvement, the position of the class boundary can be determined first, and then more aggressive elimination can be performed for the instances that are far from the boundary and less aggressive to those that are close. Some of the ideas (the "far" instances) presented in the introduction that were proposed in [20] may be useful. Another good idea is to embed instance selection in the classifier, as was discussed in the paper.

Instance selection performed during the neural network training usually allowed for achieving higher classification accuracies with a comparable number of instances. Thus for the size of datasets presented here it seems to be a good solution. On the other hand, the time of the whole process was significantly longer. It was about 3-4 times faster to perform instance selection first and then to train the network only on about 20% of selected instances. With the increase of the dataset size usually higher percentage of instances can be rejected thus making the difference even bigger. The Yeast dataset is an especially difficult problem with many classes and probably with high noise level. In that case also the *maxError* parameter did not work as expected. That was, because the network error exceeded the value of 2 for many vectors and probably the *maxError* should be set to a higher value for such datasets. However, how to determine the optimal value in such cases requires further investigations. The *minError* already realized the "far" instances idea and it worked quite well. In general a similar solution is used in Support Vector Machines, where vectors that are close to the decision border are of special consideration. In some cases, when one vector is removed as being below the *minError*, another vector that is left in its proximity should be counted with an increased coefficient while calculating the network error in order to prevent shifting of decision borders.

In summary we were able to improve the neural network classifier performance and to observe several interesting properties of the instance selection process. A further research examining all the parameters individually on a large number of datasets is probably going to help develop more effective methods.

## References

1. Abroudi, A., Shokouhifar, M., Farokhi, F.: Improving the Performance of Artificial Neural Networks via Instance Selection and Feature Dimensionality Reduction. *International Journal of Machine Learning and Computing*, Vol. 3, No. 2, April 2013
2. Antonelli, M., Ducange, P., Marcelloni, F.: Genetic training instance selection in multiobjective evolutionary fuzzy systems: A coevolutionary approach. *IEEE Transactions on Fuzzy Systems*, vol. 20(2), pp. 276-290, 2012
3. Anwar, I. M., et. al.: Instance Selection with Ant Colony Optimization. *Procedia Computer Science*, Volume 53, pp. 248–256, 2015
4. Blachnik, M., Kordos, M.: Simplifying SVM with Weighted LVQ Algorithm. *Lecture Notes in Computer Science*, Volume: 6936, pp: 212-219, IDEAL, 2011
5. Friedman, J. H., Bentley, J. L., Finkel, R. A.: An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software*, vol. 3, issue 3, pp. 209-226, 1977
6. Blake, C., Keogh, E., Merz, C.: UCI Repository of Machine Learning Databases, 1998-2015
7. The software used in the paper <http://www.kordos.com/icaisc2016>
8. Guillen, A., et. al.: New method for instance or prototype selection using mutual information in time series prediction. *Neurocomputing*, vol. 73(10-12), pp. 2030-2038, 2010
9. Hart, P.: The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, vol. 14(3), pp. 515–516, 1968
10. Hofmann, M., Klinkenberg, R.: *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. CRC Press, 2013
11. Indyk, P., Motwani, R.: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. *Proceedings of 30th Symposium on Theory of Computing*, 1988

12. Jankowski, N., Grochowski, M.: Comparison of instance selection algorithms. *Lecture Notes in Computer Science*, vol. 3070 pp. 598–603, ICAISC 2004
13. Kordos, M., Duch, W.: Variable Step Search Algorithm for Feedforward Networks, *Neurocomputing*, vol. 71, issue 13-15, pp. 2470-2480, 2008
14. Kordos, M., Duch, W.: A survey of factors influencing MLP error surface. *Control and Cybernetics*, vol. 33, No. 4, pp. 611-631, 2004
15. Leyva, E., Gonzalez, A., Perez, R.: Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognition*, vol. 48(4), pp. 1523-1537, 2015
16. Marchiori, E.: Class conditional nearest neighbor for large margin instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32(2), pp. 364-370, 2010
17. Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., Kittler, J.: A review of instance selection methods. *Artificial Intelligence Review*, vol. 34(2), pp. 133–143, 2010
18. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34(3), pp. 417-435, 2012
19. Stojanovic, M. et. al.: A methodology for training set instance selection using mutual information in time series prediction. *Neurocomputing*, vol. 141, pp. 236-245, 2014
20. Sun, X., Chan, P. K.: An Analysis of Instance Selection for Neural Networks to Improve Training Speed. *Int. Conf. on Machine Learning and Applications*, pp. 288-293, 2014
21. Wilson, D. L.: Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-2(3), pp. 408-421, 1972