2nd IEEE International Conference on Cybernetics, CYBCONF 2015, Gdynia, Poland, 24-26 June 2015, pp. 73-78

Noise Reduction in Regression Tasks with Distance, Instance, Attribute and Density Weighting

Mirosław Kordos[1] , Andrzej Rusiecki[2] , Marcin Blachnik[3]
[1] University of Bielsko-Biala, Department of Computer Science and Engineering,
Willowa 2, 43-309 Bielsko-Biała, Poland, mkordos@ath.bielsko.pl
[2] Wroclaw University of Technology, Department of Computer Engineering,
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland, andrzej.rusiecki@pwr.edu.pl
[3] Silesian University of Technology, Department of Applied Informatics,
Krasińskiego 8, 44-100 Katowice, Poland; marcin.blachnik@polsl.pl

***Abstract* – The idea presented in this paper is to gradually decrease the influence of selected training vectors on the model: if there is a higher probability that a given vector is an outlier, its influence on training the model should be limited. This approach can be used in two ways: in the input space (e.g. with such methods as k-NN for prediction and for instance selection) and in the output space (e.g. while calculating the error of an MLP neural network). The strong point of this gradual influence reduction is that it is not required to set a crisp outlier definition (outliers are difficult to be optimally defined). Moreover, according to the presented experimental results, this approach outperforms other methods while learning the model representation from noisy data.**

***Keywords* – noise reduction; instance selection; neural networks**

I. Introduction

The quality of prediction is limited by the quality of the data. Thus an important step in the data mining process should be noise reduction. The noise in the data can be generated by several factors, as imprecise or erroneousness data. The simplest solution would be to reject the wrong vectors with some instance selection method. However, that is in practice usually not the best approach for two reasons: first it would require setting a crisp rejection criteria, and second even the imprecise data contains some valuable information. Thus, here we discuss two other solutions. The first one is to softly limit the influence of the outlying points on the model: the more the point is suspected to be an outlier, the weakest is its participation in building the final model. The second solution is to correct the outlying points [11], but this requires some more knowledge, frequently the domain-specific knowledge and is not possible in every situation. That approach can also be used for increasing the data set size in cases when too few training examples could lead to model over-fitting - using interpolation we can generate additional data points. Another method to deal with over-fitting is the use of ensembles. Ensembles can reduce the variation without increasing the bias in the prediction and can allow for obtaining higher prediction accuracy than any single model [13]. Nevertheless, there are two issues with ensembles: they introduce additional complexity of the model (what may or may not be a disadvantage - depending on a given situation). And second, using ensembles in most cases prevents us from extracting comprehensive logical rules from the model (e.g. by analyzing the weights in the neural network), although some attempts have been made recently. In this paper we focus on regression tasks, because there is more research on noise reduction in classification tasks [5] and there were quite a few attempts dealing with regression.

978-1-4799-8322-3/15/$31.00 © 2015 IEEE

II. Instance Selection in Regression Tasks

There were a few attempts to reduce the influence of noise on the model in regression tasks [1], [7], [15], [19]. In this section we briefly review the most promising methods for regression problems, starting from noise reduction at the input and then discussing noise reduction at the output of the model (in this case namely the MLP neural network).

The first instance selection methods to reduce noise in classification problems - ENN (Edited Nearest Neighbor), was proposed by Wilson [21] and recently we have proposed the RegENN - an extension of this method to regression tasks [1].

Also other, more complex algorithms were developed, such as Drop1-5 [22], IB3, Gabriel Editing (GE) and Relative Neighborhood Graph Editing (RNGE), Iterative Case Filtering (ICF), ENRBF2, ELH, ELGrow and Explore [2]. A large survey including many different algorithms of instance selection for classification tasks can be found in [18] and [9].

In instance selection for classification problems, the decision about instance acceptance or rejection is very

simple. It is based on the classification results obtained usually with the k-NN algorithm. The instance can belong either to the same or to a different class as the majority of its neighbors. Comparison of the predicted class and the real class of the instance determines the decision on its acceptance/rejection. In the case of noise filters, the instances are rejected if the classes do not agree, because it indicates that the instance is an outlier.

Algorithm 1 RegENN algorithm

Require: $\mathbf{T}$
  $m \leftarrow sizeof(\mathbf{T})$;
  for $i = 1 \dots m$ **do**
    $\bar{Y}(\mathbf{x}_i)$ =Model$((\mathbf{T} \setminus \mathbf{x}_i), \mathbf{x}_i)$;
    $S \leftarrow$ k-NN$(\mathbf{T}, \mathbf{x}_i)$
    $\theta = \alpha \cdot std\left(Y\left(\mathbf{X}_S\right)\right)$
    if $\left|Y(\mathbf{x}_i) - \bar{Y}(\mathbf{x}_i)\right| > \theta$ **then**
      $\mathbf{T} \leftarrow \mathbf{T} \setminus \mathbf{x}_i$
    end if
  end for
  $\mathbf{P} \leftarrow \mathbf{T}$
  return $\mathbf{P}$

In the RegENN pseudo-code (Algorithm 1) T is the training dataset, P is the set of selected instances, $\mathbf{x}_i$ is the i-th vector, m is the number of vectors in the dataset. $Y(\mathbf{x}_i)$ is the real output value of vector $\mathbf{x}_i$, $\bar{Y}(\mathbf{x}_i)$ is the output predicted by k-NN. S is the set of k nearest neighbors of vector $\mathbf{x}_i$, $Model$ is in our case the MLP network, but it can be any model used for the final prediction. The k-NN algorithm is used to determine the subset S of k closest neighbors to $\mathbf{x}_i$. Θ is the rejection threshold (it will be explained in detail later) and α is a certain coefficient, in our case α=5. $std\left(Y\left(\mathbf{X}_S\right)\right)$ is the standard deviation of the outputs of the vectors in S.

III. Attribute Weighting

The first step of our data preprocessing is standardization and all the processes described in this paper are performed on standardized data. That allows for not dealing with the scaling problems (e.g. for replacing correlation by covariance) and for easy result comparison. We propose to use the weighting scheme for attribute, distance, density and instance weighting, based on the exponential function. The final weight w of each point contribution in the k-NN used for instance selection, can be imagined as multiplication of the four weights. Although the weights are not literally multiplied, but more complex operations are performed, which, however, can be conceptually understood in this way (Eq. 1).

$$w = w_{attr} \cdot w_{dist} \cdot w_{dens} \cdot w_{out} \tag{1}$$

When we use an MLP neural network as the inner instance selection algorithm [1], the network requires only the density and instance weighting. However, additional attribute and distance weighting allowed us to achieve with k-NN the performance of an MLP network, while making the whole instance selection process simpler.

The simplest version of attribute weighting is feature selection, where the features that do not improve prediction are rejected from the feature set. However, the experiments conducted by several authors indicated that a better results can be obtained by feature weighting [20], where the more important features are assigned larger weights. Therefore we use the standard technique of attribute weighting by their correlation with the output. The j-th attribute weight w_j equals to that attribute correlation or covariance for standardized data with the output (Eq. 2):

$$w_j = \frac{1}{N-1} \sum_{i=1}^{N} (x_{ij} - \overline{x}_j)(y_i - \overline{y}) \tag{2}$$

Thus w_j it is the covariance between the attribute x_j and the output value y transposed by the k-NN decay function (see Fig. 1), where N is the number of instances (N equals either the number of instances in the data set or $N = k$ if applied locally to the k nearest neighbors) and b is a coefficient (we use b = 0.5).

Another issue is that the globally optimal set of feature weights may be different than the locally optimal set of weights within the k nearest neighbors of a given point. The first set can be used to determine the k nearest neighbors and the second set to predict the value of the point of interest. Though it is possible to generate such artificial data that the two sets of weights will be so different that this approach will be unstable, we did not observe significant differences between the two sets of attribute weights on real world datasets.

Then the weights are used to calculate the distance $d1$ between two vectors $x1$ and $x2$ (Eq. 3):

$$d1(x1, x2) = \left(\frac{1}{\sum_{j=1}^{F} w_j} \sum_{j=1}^{F} w_j (x1_j - x2_j)^p \right)^{\frac{1}{p}} \tag{3}$$

where F is the number of features and p is the exponent in the Minkovsky distance measure. In the experiments we used $p = 2$.

IV. Distance Weighting

The results obtained with k-NN significantly depend and the value of k and on the weighting scheme [12]. The idea of distance weighting in k-NN is not new and several approaches have already been proposed [6]. We perform distance weighting using the same k-NN decay function (Eq. 4):

$$d2(x1, x2) = Exp\left(-b(d1(x1, x2))^2\right) \tag{4}$$

V. Density Weighting

Density weighting is specific to instance selection in regression tasks and it has neither application to k-NN based prediction nor to instance selection for classification tasks. The instance acceptance/rejection threshold Θ reflects the similarity of two instances. But, as the experiments showed, better results in terms of prediction accuracy and compression can be achieved if variable Θ is used. Then the threshold is expressed rather not in absolute values but in percentage of the standard deviation of the output of k nearest neighbors $std\,(Y\,(\mathbf{X}_S))$. Thus, in the areas of dense data the standard deviation of the k nearest neighbor for instance can be 0.3, while in the sparse zones it can be for instance 3.0. So the threshold Θ should be adjusted accordingly as $\Theta = \alpha \cdot std\,(Y\,(\mathbf{X}_S))$ (Eq. 5). Lower α always leads to better data compression, because fewer instances are selected. On the other hand, in terms of prediction accuracy, there exists an optimal value of α, which according to our experiments is usually between 5 and 8 for noise reduction algorithms as the RegENN. If the difference between the value predicted by k-NN (or another algorithm) and the actual value is greater than Θ then the vector is rejected.

$$d3(x1, x2) = \alpha \cdot std\,(Y\,(\mathbf{X}_S)) \cdot d2(x1, x2) \qquad (5)$$

VI. Instance Weighting

Instance weighting is especially important in case of noisy data. The simplest version of instance weighting is instance selection. We propose that instance weighting should be used even inside the instance selection.

The problem with crisp instance selection i.e. rejecting or accepting an instance in the training set is to define the optimal Θ. In many cases it is not clear what θ value should be used. One solution is to optimize Θ in crossvalidation. However, as the experiments showed, this is a similar problem as with feature selection and a similar solution seems to be most effective, i.e. assigning a weight to each instance instead of crisp rejection threshold. Only instances for which the weight is really very small should be totally rejected. The rationale behind this is that even the partially noisy instances still carry some useful information.

In our work [17] we evaluated several approaches to noise reduction in MLP neural network training. These approaches could be divided into two groups: input noise reduction and output noise reduction. We evaluated input noise reduction by the RegENN algorithm and by assigning each vector a weight with a modified Global Anomaly Score algorithm and then multiplying the error the network made on that vector by its anomaly score. (The anomaly score is proportional to the average distance between a given point and its k nearest neighbors.) The output noise reduction methods were based on modification of the network error function, to make the learning process less sensitive to outliers. In this case we didn't use any information about the vectors obtained before the network training. Instead of applying MSE error, we used other functions ,as proposed in Several such error functions were proposed in [3], [4], [14], which reduced the influence of the most "abnormal" vectors on the total network error.

In this paper, we discuss another very simple solution which in the tests performed especially well: a product of the standard MSE function and the same exponential function we used for k-NN weighting in k-NN based prediction and instance selection, given in the formula (Eq. 6) and presented in Fig. 1.

$$E = (y_a - y_p)^2 \cdot Exp(-a(y_a - y_p)^2) \qquad (6)$$

where y_a is the actual and y_p is the predicted output. The parameter a is gradually increased during the network training. When the training starts with random network weights, the network produces random errors for each vector. So at this stage we should not interfere in the learning process, thus a must be small. As the network learns the data representation, it starts to response with higher errors to the outliers and we can increase a to reduce the outlier influence on the network mappings. The error first increases as the distance between the actual vector output and the value predicted by the network increases. Then it begins to increase slower and finally to decrease asymptotically to zero. In this way the bigger the outlier is, the weaker is its influence on the network mappings. That works very well in practice.

We can also use the instance weighting with k-NN on noisy data. In that case we first determine with k-NN the instance weights $w_i = Exp(-b(y_a - y_p)^2)$ for each vector and then, we run the k-NN algorithm the second time, now applying the weights. That reduces the outlier influence on the final prediction in an analogical way like with the MLP network.

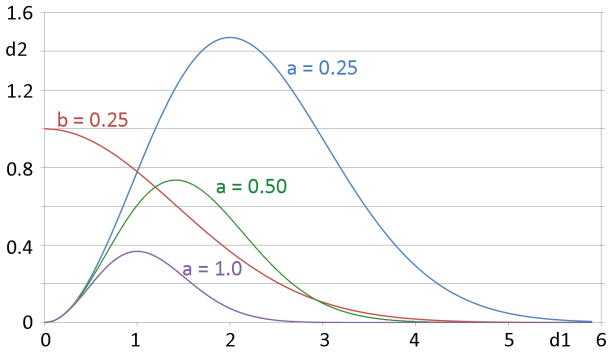

Figure 1. k-NN decay function (b=0.25) and MLP Error functions for a = 0.25, 0.50, 1.0.

VII. Experimental Results

We performed the tests with linear and exponential weighting functions in three areas:

1) k-NN based prediction;
2) RegENN-based instance selection with MLP network as the prediction;
3) MLP - based prediction with output weighting;

All the tests were carried out on the same five regression datasets. We used the datasets in their original form and with different amounts of noise added to the inputs as well as to the output value of the training dataset, as listed below:

0) $p = 0$, $f = 0$;
1) $p = 0.5$, $f = 0.20$;
2) $p = 0.8$, $f = 0.25$;
3) $p = 1.5$, $f = 0.30$;
4) $p = 2.5$, $f = 0.35$;
5) $p = 4.0$, $f = 0.40$;

where p (amplitude) is the standard deviation of the Gaussian distribution from which the random noise was added (the noise amplitude was each time a random number from zero to p) and f (frequency) is the probability (with uniform distribution) of the noise being added to a given data point. The first line in tables 1-5 represents the input noise level and the output noise level. Thus 0/0 means that no noise was added, and e.g. 2/0 means that only the input noise was added with p=0.8 and f=0.25, 0/2 indicates that only output added with p=0.8 and f=0.25, and 3/3 means that input and output noise was added, both with p=1.5 and f=0.30.

We evaluated performance of the following algorithms:

1) kNN;
2) AD-kNN (k-NN with Attribute and Distance weighting);
3) ADI-kNN (k-NN with Attribute, Distance and Instance weighting);
4) MLP-MSE (MLP network trained with MSE error function);
5) RegENN MLP-MSE (MLP network trained with MSE error function on the data selected by RegENN weighted by attribute and distance);
6) RegWENN MLP-MSE (MLP network trained with MSE error function on the data selected by RegENN weighted by attribute, distance, outlier and density);
7) MLP-LTA (MLP network trained with Least Trimmed Absolute Values error function [16]);
8) MLP-EXP (MLP network trained with the multiplication of Exponential and MSE error function);

Although the optimal k was lower for lower noise levels (eg. k=7) and higher for higher noise levels (eg. k=30), we kept k=11 during all experiments. The MLP trained with the LTA error function [16] was added to comparison, because our previous tests with various noise-robust MLP training methods showed the LTA obtained the best performance on noisy data. All the tests were run in 10-fold crossvalidation. We used a single hidden layer MLP network with 6 hidden neurons for each dataset, with hyperbolic tangent transfer functions in the hidden layer and linear transfer function in the output layer. We trained the network using 12 iterations of VSS algorithm [10].

We present the results obtained for five datasets. The upper rows in the tables contains the average RMSE value and the lower rows the RMSE standard deviation. Figure 2 shows the weighted average results over the five datasets for different noise levels. To make the error values comparable, the weights were 3 for Mortgage (because of its very low error vales) and 1 for each other dataset. The software used in the experiments was created in C# and it can be obtained together with the datasets from *www.kordos.com/cybconf2015*.

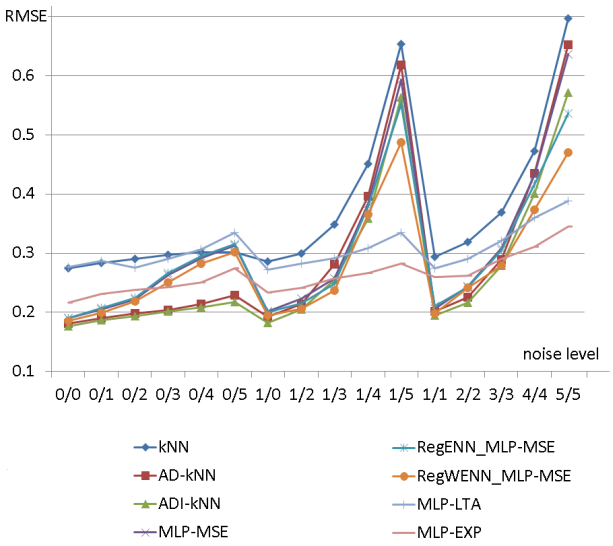

Figure 2. Weighted average prediction results over the five datasets for different noise levels.

VIII. Conclusions

The first conclusion is that one should always use k-NN with feature weighting. Although it is not our discovery, the improvement due to feature weighting is so enormous that we must mention this here. All the other weightings added to k-NN (distance and outlier weighting) and to RegENN (density weighting) also bring some improvement, although usually not so big as feature weighting. Outlier weighting (instance weighting) with the exponential function brings some improvement to k-NN and to RegENN based on that k-NN, but the improvement is relatively small. On the other hand, the outlier weighting applied in an analogical way to the MLP network error function performs really very well on noisy data, allowing for obtaining much lower prediction errors. What is interesting, density weighting applied to RegENN reduces the errors of the final predictor (MLP network

Table I. RMSE IN 10-FOLD CROSSVALIDATION FOR BUILDING DATASET. THE UPPER ROWS IN EACH TABLE CONTAINS THE AVERAGE RMSE VALUE AND THE LOWER ROWS STANDARD DEVIATION.

Method	0/0	0/1	0/2	0/3	0/4	0/5	1/0	2/0	3/0	4/0	5/0	1/1	2/2	3/3	4/4	5/5
k-NN	0.37	0.38	0.40	0.41	0.43	0.43	0.37	0.39	0.42	0.53	0.72	0.39	0.43	0.47	0.57	0.80
	0.04	0.04	0.05	0.05	0.05	0.04	0.04	0.04	0.04	0.06	0.04	0.05	0.05	0.05	0.06	0.06
AD-kNN	0.25	0.26	0.27	0.27	0.30	0.35	0.26	0.27	0.34	0.46	0.69	0.27	0.28	0.35	0.58	0.72
	0.02	0.02	0.02	0.02	0.03	0.06	0.02	0.02	0.02	0.01	0.07	0.02	0.03	0.03	0.07	0.07
ADI-kNN	0.25	0.26	0.27	0.28	0.30	0.33	0.24	0.28	0.33	0.42	0.68	0.27	0.29	0.36	0.59	0.70
	0.02	0.02	0.02	0.03	0.03	0.04	0.02	0.02	0.02	0.04	0.05	0.02	0.03	0.03	0.06	0.07
MLP-MSE	0.25	0.26	0.30	0.34	0.40	0.45	0.25	0.27	0.33	0.49	0.64	0.27	0.30	0.40	0.53	0.76
	0.02	0.01	0.02	0.03	0.03	0.02	0.01	0.02	0.02	0.03	0.10	0.01	0.01	0.03	0.06	0.05
RegENN MLP-MSE	0.25	0.26	0.28	0.31	0.40	0.42	0.25	0.28	0.32	0.47	0.72	0.28	0.32	0.40	0.53	0.67
	0.02	0.02	0.02	0.03	0.03	0.05	0.02	0.01	0.02	0.02	0.05	0.01	0.02	0.03	0.04	0.08
RegWENN MLP-MSE	0.25	0.26	0.28	0.29	0.38	0.41	0.24	0.28	0.30	0.43	0.71	0.26	0.31	0.36	0.47	0.61
	0.02	0.01	0.02	0.02	0.03	0.05	0.02	0.01	0.02	0.02	0.05	0.01	0.03	0.03	0.04	0.06
MLP-LTA	0.45	0.44	0.43	0.46	0.49	0.52	0.44	0.42	0.45	0.45	0.50	0.44	0.45	0.48	0.54	0.60
	0.03	0.02	0.02	0.01	0.02	0.03	0.01	0.02	0.02	0.02	0.04	0.03	0.01	0.03	0.03	0.03
MLP-EXP	0.24	0.27	0.58	0.27	0.30	0.32	0.29	0.29	0.30	0.34	0.35	0.31	0.35	0.38	0.42	0.50
	0.02	0.04	0.12	0.07	0.04	0.03	0.11	0.10	0.04	0.06	0.03	0.09	0.10	0.05	0.05	0.09

Table II. RMSE IN 10-FOLD CROSSVALIDATION FOR STEEL DATASET.

Method	0/0	0/1	0/2	0/3	0/4	0/5	1/0	2/0	3/0	4/0	5/0	1/1	2/2	3/3	4/4	5/5
kNN	0.30	0.30	0.31	0.32	0.34	0.34	0.34	0.39	0.47	0.70	0.37	0.40	0.45	0.55	0.68	0.74
	0.09	0.09	0.10	0.10	0.09	0.08	0.09	0.09	0.09	0.07	0.06	0.10	0.09	0.08	0.07	0.07
AD-kNN	0.28	0.29	0.28	0.30	0.31	0.32	0.29	0.39	0.47	0.67	0.32	0.36	0.39	0.51	0.61	0.71
	0.08	0.08	0.08	0.08	0.09	0.08	0.07	0.08	0.09	0.07	0.10	0.10	0.09	0.09	0.10	0.08
ADI-kNN	0.27	0.28	0.28	0.30	0.32	0.32	0.29	0.35	0.47	0.65	0.33	0.35	0.40	0.47	0.58	0.68
	0.09	0.09	0.09	0.10	0.09	0.08	0.09	0.08	0.08	0.10	0.06	0.08	0.09	0.08	0.09	0.07
MLP-MSE	0.28	0.29	0.32	0.36	0.42	0.50	0.28	0.31	0.38	0.51	0.66	0.28	0.35	0.42	0.53	0.70
	0.08	0.11	0.08	0.08	0.08	0.07	0.09	0.08	0.08	0.06	0.06	0.08	0.07	0.07	0.08	0.12
RegENN MLP-MSE	0.27	0.29	0.33	0.36	0.42	0.50	0.30	0.33	0.35	0.52	0.67	0.29	0.33	0.41	0.55	0.68
	0.07	0.07	0.08	0.06	0.07	0.10	0.09	0.07	0.10	0.09	0.08	0.07	0.07	0.09	0.08	0.10
RegWENN MLP-MSE	0.27	0.29	0.33	0.35	0.42	0.48	0.30	0.32	0.35	0.52	0.65	0.30	0.33	0.40	0.55	0.65
	0.08	0.08	0.08	0.08	0.07	0.12	0.10	0.08	0.09	0.09	0.09	0.07	0.07	0.10	0.12	0.13
MLP-LTA	0.42	0.42	0.43	0.45	0.49	0.52	0.40	0.46	0.47	0.49	0.48	0.44	0.44	0.48	0.52	0.55
	0.03	0.04	0.04	0.03	0.04	0.04	0.03	0.05	0.05	0.03	0.03	0.04	0.03	0.03	0.04	0.06
MLP-EXP	0.44	0.45	0.48	0.50	0.53	0.60	0.45	0.41	0.41	0.55	0.55	0.46	0.69	0.54	0.58	0.65
	0.13	0.14	0.17	0.12	0.11	0.15	0.11	0.12	0.10	0.16	0.18	0.12	0.22	0.16	0.18	0.15

Table III. RMSE IN 10-FOLD CROSSVALIDATION FOR CRIME DATASET.

Method	0/0	0/1	0/2	0/3	0/4	0/5	1/0	2/0	3/0	4/0	5/0	1/1	2/2	3/3	4/4	5/5
kNN	0.58	0.59	0.60	0.61	0.61	0.60	0.58	0.57	0.62	0.70	0.83	0.59	0.60	0.64	0.74	0.90
	0.07	0.08	0.07	0.08	0.09	0.08	0.07	0.07	0.09	0.08	0.09	0.07	0.08	0.08	0.05	0.11
AD-kNN	0.58	0.59	0.60	0.60	0.59	0.59	0.59	0.59	0.62	0.69	0.85	0.59	0.61	0.63	0.70	0.89
	0.09	0.09	0.07	0.07	0.07	0.08	0.09	0.08	0.08	0.07	0.14	0.08	0.09	0.10	0.07	0.10
ADI-kNN	0.58	0.59	0.59	0.61	0.60	0.60	0.59	0.60	0.61	0.68	0.80	0.59	0.61	0.62	0.67	0.83
	0.09	0.08	0.08	0.08	0.07	0.07	0.08	0.08	0.06	0.06	0.18	0.09	0.09	0.08	0.11	0.15
MLP-MSE	0.58	0.60	0.61	0.60	0.62	0.61	0.59	0.59	0.64	0.75	1.20	0.60	0.61	0.65	0.70	1.05
	0.08	0.08	0.07	0.06	0.09	0.06	0.10	0.08	0.11	0.10	0.15	0.08	0.08	0.07	0.11	0.12
RegENN MLP-MSE	0.57	0.60	0.59	0.62	0.61	0.66	0.61	0.64	0.66	0.76	1.10	0.58	0.62	0.65	0.79	0.89
	0.08	0.09	0.07	0.08	0.09	0.09	0.07	0.08	0.09	0.10	0.20	0.09	0.07	0.10	0.06	0.13
RegWENN MLP-MSE	0.57	0.60	0.59	0.61	0.61	0.63	0.61	0.60	0.66	0.76	1.04	0.58	0.61	0.62	0.74	0.84
	0.08	0.08	0.08	0.08	0.09	0.08	0.07	0.08	0.10	0.10	0.23	0.09	0.08	0.09	0.08	0.12
MLP-LTA	0.68	0.68	0.69	0.67	0.70	0.70	0.69	0.68	0.70	0.71	0.72	0.68	0.70	0.71	0.72	0.72
	0.06	0.05	0.05	0.06	0.04	0.06	0.05	0.05	0.06	0.06	0.05	0.06	0.03	0.07	0.03	0.07
MLP-EXP	0.60	0.61	0.63	0.63	0.62	0.60	0.64	0.65	0.64	0.61	0.61	0.63	0.62	0.61	0.61	0.64
	0.10	0.10	0.07	0.11	0.08	0.10	0.09	0.09	0.09	0.08	0.08	0.10	0.08	0.11	0.09	0.08

in our case), but what it reduces more is the number of selected instances. We did not report the number here, but on average the best results in the experimental evaluation were obtained when rejecting about 10-15% of instances from the training set without density weighting. With density weighting the number increased to 20-30%. While in classification tasks, frequently even over 70% of instances can be rejected, in regression it is impossible, because each point in the data space matters, not only the locations of the class boundaries. The final conclusion is that the distance and instance (outlier) weighting with exponential function brings some improvement to the k-NN and RegENN algorithms on noisy data in regression tasks and that it brings a very significant improvement when applied to MLP network error function. It is likely that the best results can be obtained by combining weighted RegENN with EXP. However, we have not verified this experimentally yet.

Table IV. RMSE IN 10-FOLD CROSSVALIDATION FOR YACHT HYDRODYNAMICS DATASET.

Method	0/0	0/1	0/2	0/3	0/4	0/5	1/0	2/0	3/0	4/0	5/0	1/1	2/2	3/3	4/4	5/5
kNN	0.64	0.65	0.65	0.67	0.66	0.67	0.65	0.65	0.67	0.71	0.86	0.67	0.67	0.72	0.79	1.00
	0.18	0.19	0.19	0.19	0.20	0.20	0.18	0.18	0.18	0.19	0.17	0.17	0.18	0.18	0.25	0.16
AD-kNN	0.14	0.15	0.18	0.22	0.24	0.26	0.14	0.17	0.28	0.45	0.70	0.17	0.19	0.27	0.48	0.85
	0.04	0.03	0.10	0.09	0.09	0.10	0.03	0.05	0.07	0.12	0.13	0.05	0.06	0.08	0.09	0.14
ADI-kNN	0.11	0.13	0.16	0.20	0.21	0.21	0.11	0.14	0.22	0.36	0.64	0.13	0.20	0.28	0.46	0.78
	0.04	0.06	0.07	0.11	0.08	0.11	0.04	0.03	0.05	0.03	0.18	0.05	0.06	0.08	0.11	0.13
MLP-MSE	0.21	0.24	0.26	0.46	0.52	0.60	0.27	0.29	0.28	0.52	0.88	0.23	0.36	0.49	0.80	1.14
	0.06	0.07	0.08	0.08	0.14	0.11	0.11	0.19	0.08	0.13	0.15	0.05	0.14	0.13	0.07	0.19
RegENN MLP-MSE	0.22	0.28	0.35	0.53	0.56	0.62	0.25	0.20	0.32	0.67	0.98	0.28	0.36	0.53	0.76	1.10
	0.06	0.09	0.11	0.10	0.15	0.12	0.06	0.04	0.05	0.25	0.12	0.08	0.08	0.12	0.16	0.10
RegWENN MLP-MSE	0.20	0.21	0.30	0.48	0.50	0.60	0.22	0.19	0.30	0.50	0.78	0.21	0.36	0.49	0.68	0.89
	0.07	0.09	0.09	0.08	0.10	0.10	0.05	0.03	0.06	0.12	0.14	0.09	0.08	0.10	0.13	0.13
MLP-LTA	0.38	0.45	0.39	0.46	0.45	0.58	0.35	0.43	0.39	0.47	0.53	0.37	0.43	0.52	0.63	0.69
	0.08	0.06	0.06	0.09	0.10	0.10	0.04	0.08	0.07	0.06	0.09	0.08	0.10	0.13	0.12	0.10
MLP-EXP	0.39	0.45	0.48	0.45	0.42	0.57	0.71	0.70	0.22	0.47	0.53	0.56	0.46	0.59	0.64	0.68
	0.10	0.12	0.14	0.15	0.10	0.12	0.23	0.26	0.09	0.22	0.20	0.18	0.20	0.21	0.11	0.13

Table V. RMSE IN 10-FOLD CROSSVALIDATION FOR MORTGAGE DATASET.

Method	0/0	0/1	0/2	0/3	0/4	0/5	1/0	2/0	3/0	4/0	5/0	1/1	2/2	3/3	4/4	5/5
kNN	0.080	0.088	0.092	0.091	0.094	0.096	0.104	0.132	0.218	0.381	0.702	0.103	0.141	0.218	0.371	0.690
	0.013	0.005	0.016	0.017	0.030	0.080	0.009	0.014	0.022	0.032	0.050	0.009	0.014	0.022	0.033	0.051
AD-kNN	0.073	0.082	0.085	0.085	0.090	0.094	0.095	0.132	0.222	0.371	0.670	0.103	0.136	0.238	0.382	0.684
	0.012	0.011	0.021	0.030	0.060	0.014	0.010	0.029	0.041	0.077	0.009	0.010	0.029	0.041	0.080	0.090
ADI-kNN	0.073	0.083	0.085	0.085	0.088	0.090	0.088	0.115	0.185	0.329	0.572	0.096	0.113	0.204	0.301	0.519
	0.011	0.015	0.029	0.050	0.081	0.011	0.015	0.029	0.051	0.078	0.011	0.015	0.029	0.051	0.076	0.110
MLP-MSE	0.079	0.095	0.102	0.123	0.130	0.133	0.081	0.119	0.162	0.274	0.461	0.092	0.124	0.177	0.300	0.457
	0.013	0.014	0.033	0.043	0.035	0.013	0.014	0.033	0.043	0.061	0.058	0.014	0.033	0.043	0.060	0.102
RegENN MLP-MSE	0.076	0.090	0.097	0.120	0.129	0.134	0.082	0.109	0.122	0.222	0.302	0.085	0.107	0.154	0.239	0.316
	0.010	0.012	0.024	0.025	0.041	0.010	0.015	0.030	0.041	0.055	0.042	0.010	0.022	0.038	0.056	0.081
RegWENN MLP-MSE	0.074	0.093	0.101	0.111	0.130	0.120	0.081	0.102	0.113	0.268	0.220	0.090	0.115	0.135	0.211	0.268
	0.010	0.015	0.021	0.027	0.048	0.012	0.012	0.031	0.037	0.048	0.039	0.011	0.024	0.032	0.071	0.082
MLP-LTA	0.084	0.095	0.088	0.092	0.108	0.119	0.087	0.102	0.114	0.129	0.143	0.089	0.099	0.125	0.149	0.165
	0.012	0.011	0.019	0.014	0.018	0.014	0.011	0.019	0.013	0.022	0.010	0.014	0.014	0.015	0.022	0.024
MLP-EXP	0.083	0.083	0.093	0.102	0.123	0.140	0.080	0.082	0.095	0.108	0.125	0.089	0.105	0.105	0.132	0.147
	0.012	0.013	0.018	0.012	0.015	0.012	0.013	0.018	0.012	0.020	0.012	0.013	0.018	0.012	0.020	0.020

REFERENCES

[1] Blachnik, M., Kordos, M., Instance Selection in Logical Rule Extraction for Regression Problems. Lecture Notes in Artificial Intelligence, vol. 8468, pp.: 40-51, ICAISC, June 2014

[2] Cameron-Jones, R.M., Instance selection by encoding length heuristic with random mutation hill climbing. The Eighth Australian Joint Conference on Artificial Intelligence. pp. 99-106, 1995

[3] El-Melegy, M., Essai, M., Ali, A., Robust Training of Artificial Feedforward Neural Networks. Studies in Computational Intelligence, vol. 201, pp. 217-242. Springer, 2009

[4] El-Melegy, M., Random Sampler m-estimator Algorithm with Sequential Probability Ratio Test for Robust Function Approximation via Feed-forward Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, 24(7), pp. 1074-1085, 2013

[5] Garcia, S., Luengo, J., Herrera, F.: Data Preprocessing in Data Mining, Springer, 2014

[6] Gou J., et. al., A New Distance-weighted k-nearest Neighbor Classifier. Journal of Information & Computational Science 9:6, pp. 1429–1440, 2012

[7] Guillen, A., Applying Mutual Information for Prototype or Instance Selection in Regression Problems. ESANN 2009

[8] Hampel, F.R., et. al., Robust Statistics: The Approach Based on Influence Functions, Wiley-Interscience, New York, April 2005

[9] Jankowski, N., Grochowski, M., Comparison of Instance Selection Algorithms. Lecture Noter in Computer Science, pp. 580-585 and 598-603, June 2004

[10] Kordos, M., Duch, W., A survey of factors influencing MLP error surface. Control and Cybernetics, vol 33, issue: 4, pp. 611-631, 2004

[11] Kordos, M., Blachnik, M., et. al., A Hybrid System with Regression Trees in Steel-Making Process. Lecture Notes in Artificial Intelligence, vol. 6678, pp. 222-230, HAIS, May 2011

[12] Kordos, M., Strzempa, D., Blachnik, M., Do We Need Whatever More than k-NN? Lecture Notes in Artificial Intelligence, vol. 6113, pp. 414-421, ICAISC, June 2010

[13] Kuncheva, L.I., Combining Pattern Classifiers: Methodsand Algorithms. Wiley, 2014

[14] Olive, D.J., Hawkins, D.M., Robustifying Robust Estimators, 2007

[15] Rodriguez-Fdez, I., et. al., An Instance Selection Algorithm for Regression and its Application in Variance Reduction. 2013 IEEE International Conference on Fuzzy Systems, pp. 1–8, July 2013

[16] Rusiecki, A., Robust Learning Algorithm Based on LTA Estimator. Neurocomputing, vol. 120, pp. 624-632, 2013

[17] Rusiecki, A., Kordos, M., et. al., Training Neural Networks on Noisy Data. Lecture Notes in Artificial Intelligence, vol. 8467, pp: 131-142, ICAISC, June 2014

[18] Salvador, G., Derrac, J., Ramon, C., Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. IEEE Transactions on Pattern Analysis and Machine Intelligence 34, pp. 417-435, 2012

[19] Tolvi., J., Genetic Algorithms for Outlier Detection and Variable Selection in Linear Regression Models. Soft Computing, 8(8), pp. 527–533, 2004

[20] Wettschereck, et. al., A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. Artificial Intelligence Review, vol. 11, issue 1-5, pp. 273-314, 1997

[21] Wilson, D., Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. on Systems, Man, and Cybernetics, pp. 408-421, 1972

[22] Wilson, D., Martinez, T., Reduction Techniques for Instance-based Learning Algorithms. Machine Learning, Vol. 38, pp. 251-268, 2000