Training Neural Networks on Noisy Data

Andrzej Rusiecki[1], Mirosław Kordos[2], Tomasz Kamiński[2], and Krzysztof Greń[2]

[1] Wroclaw University of Technology, Institute of Computer Engineering, Control and Robotics, Wrocław, Wybrzeże Wyspiańskiego 27, Poland
andrzej.rusiecki@pwr.wroc.pl
[2] University of Bielsko-Biala, Department of Mathematics and Computer Science, Bielsko-Biała, Willowa 2, Poland
mkordos@ath.bielsko.pl

Abstract. This paper discusses approaches to noise-resistant training of MLP neural networks. We present various aspects of the issue and the ways of obtaining that goal by using two groups of approaches and combinations of them. The first group is based on a different processing of each vector depending of the likelihood of the vector being an outlier. The likelihood is determined by instance selection and outlier detection. The second group is based on training MLP neural networks with non-differentiable robust objective functions. We evaluate the performance of particular methods with different level of noise in the data for regression problems.

1 Introduction

Multilayer perceptrons (MLP) are among the most popular approaches used to build data-based models for various applications. They are usually considered as reliable and easy-to-use tools. However, their performance strongly depends on the quality of the training data [3, 18]. In this paper we present and test some state of the art methods, which allows training the MLPs on contaminated datasets.

MLP networks are trained by minimizing an error function on the training set, to make the network map the input data distribution to output space variables, which in case of regression are real numbers. However, since the error is minimized to make the network output for each vector as close as possible to the real vector output, it is crucial that the training data is of a good quality. Good quality means that the data reflects the underlying problem. If the data contains a lot of faulty measurements, other errors and outliers it obviously does not match the problem well, so also the neural network trained on that data will not.

In this paper we take into account two groups of methods to deal with the noisy data problem. The first group of methods makes some adjustment to other neural network itself, such as the error function, the neuron transfer functions and others to make the network to process differently data points of different properties in such a way that it is less sensitive to outliers. These methods are presented in section 2.

The second group uses outlier reduction methods, which are applied to the data prior to the network training. Thus, the data is modified and a typical MLP network is then trained on that data. This is discussed in section 3.

L. Rutkowski et al. (Eds.): ICAISC 2014, Part I, LNAI 8467, pp. 131–142, 2014.

Finally we discuss the possibilities of joining the two groups of methods together. The section 4 presents the experimental comparison of nine different methods from all the three groups on seven regression tasks performed with various amount of noise added to the data. Finally the section 5 concludes this work.

2 Data with Outliers and Robust Learning

An outlier can be defined as an observation distant from the bulk of the data. Such observations may be caused by human mistakes, measurement or rounding errors, long-tailed noise, etc. This is why outliers are usually considered as gross errors but they can be also potentially meaningful. In typical raw data, the quantity of outliers ranges from 1% to 10% [11], however it is hard to predict how much outliers the data contain.

The feedfoward neural network trained to minimize MSE (mean squared error) builds a model based on fitting training patterns as close as possible (according to the MSE measure). Such approach is indeed optimal for data contaminated by errors generated from zero-mean Gaussian distribution but when outliers appear in the training set, the network model becomes unreliable [3, 18, 19]. This is why several robust learning algorithms, to train neural networks on the data with outliers, have been proposed [3–5, 18, 25]. Such methods, usually based on the robust statistical estimators, should be reliable also when the training data quality is unknown.

One of the basic approaches to make a learning algorithm more robust to outliers is to replace the MSE performance measure by another function. In this approach, the robustness to outliers is achieved by reducing the impact of large training residuals, potentially caused by outlying data points. Many such functions derived from robust statistical estimators can be found in the literature. New LMLS (Least Mean Log Squares) error function was proposed by Liano [18]. Chen and Jain [3] applied the Hampel's hyperbolic tangent with scale estimator β, determining residuals suspected to be caused by outliers, Chunag and Su [4] added the annealing scheme to decrease the value of β. Error functions based on the tau-estimators [19] and the MCD (Minimum Covariance Determinant) [24] were also proposed. El-Melegy *et al.* presented the Simulated Annealing for Least Median of Squares (SA-LMedS) algorithm [5], while Rusiecki proposed the LTS (Least Trimmed Squares) [23] and LTA (Least Trimmed Absolute Values) [26] algorithms. The RANSAC (random sample consensus) framework, known from the area of image processing, was applied to the MLPs learning by El-Melegy [6–8].

2.1 Trimmed and Median-Based Error Measures

In the previous research many modified performance functions have been examined and the best results have been obtained with the quantile-based and trimmed performance measures [5, 17, 25, 26]. Trimmed and quantile-based robust estimators are proved to be outlier-resistant, so it is not surprising that they perform well also in network training.

The main problem is that such measures are not continuous and some approximations of their derivatives in gradient-based learning must be used. An alternative approach is to train the network with non-gradient methods. In this paper we use the Variable Step

Search (VSS) Algorithm [14] to train the network with robust non-differentiable error measures. The main idea of the VSS algorithm is to guess the optimal modifications of single weights at each iteration based on their changes in previos iterations and then to adjust the changes. Since change of a single weight does not change signal propagations in the entire network, the signals (unlike in gradent-based methoods) are propagated each time only through the recently changed fragments of the network. However, we do not focus on the learning algorithm itself and use VSS with the same parameters through all the tests. It is also not crucial to use VSS and it can be replaced with several other MLP training methods.

3 LTA and ILMedS Algorithms

One of the desired properties of robust estimators is a high breakdown point. It is defined as the smallest percentage ϵ^* of contaminated data that can cause the estimator to take on aberrant values [11]. Theoretically, for the least squares method the breakdown point $\epsilon^* = 0$. The least trimmed absolute value (LTA) and the least median of squares (LMedS) are known in the robust statistics to be the classical high breakdown point robust estimators (breakdown point close to $\epsilon^* = 0.5$). In fact, the breakdown point $\epsilon^* = 0.5$ is the best that can be expected from any estimator [22]. Unlike robust M-estimators, the LTA and LMedS do not change operations performed on single residuals (such as squaring or taking absolute value), but replace the sum of residuals with a trimmed sum or a certain statistical value as median. Hence, the LMedS estimator is based on the Chebyshev (L_∞) norm and the LTA is a trimmed version of L_1 norm.

3.1 Least Trimmed Absolute Values

The least trimmed absolute value estimator (LTA) is one of the well-known robust location estimators. Similarly to the least trimmed squares (LTS) [22] it does not change operations performed on residuals. Hence, in this case, residuals are not squared but their absolute values are taken. Then the summation is replaced with a trimmed sum.

Let us consider the general nonlinear regression model:

$$y_i = \eta(x_i, \theta) + \epsilon_i, \quad i = 1, \ldots, n, \tag{1}$$

where y_i denotes the dependent variable, $x_i = (x_{i1}, \ldots, x_{ik})$ the independent input vector, $\theta \in R^p$ is the underlying parameter vector, and ϵ_i denotes independent and identically distributed (iid) random errors with a continuous distribution function. Now we can define the least trimmed absolute value estimator:

$$\hat{\theta} = \arg \min_{\theta \in R^p} \sum_{i=1}^{h} (|r|)_{i:n}, \tag{2}$$

where $(|r|)_{1:n} \leq \cdots \leq (|r|)_{n:n}$ are the absolute residuals $|r_i(\theta)| = |y_i - \eta(x_i, \theta)|$ sorted in ascending order. In the summation only h smallest absolute values of the residuals are used. Setting the trimming constant h as $n/2 < h \leq n$ we can decide what percentage of largest residuals will not affect the estimator.

LTA Error Criterion. The new robust error criterion based on the LTA estimator was introduced in [26] as:

$$E_{LTA} = \sum_{i=1}^{h} (|r|)_{i:n}, \tag{3}$$

where $(|r|)_{1:n} \leq \cdots \leq (|r|)_{n:n}$ are ordered absolute network output residuals for each training pattern.

This error measure should provide robustness to outliers excluding from the training process patterns causing largest errors (assuming that these patterns are outliers). The trimming constant h can be set empirically but in [26] a simple approach to estimate the scaling factor was proposed. Calculation of h is based on a robust measure of scale, namely the median of all absolute deviations from the median (MAD)[13]:

$$\text{MAD}\ (r_i) = 1.483\ \text{median}|r_i - median(r_i)|. \tag{4}$$

The trimming parameter is then calculated as:

$$h = \|\{r_i : |r_i| < 3 * \text{MAD}(|r_i|), i = 1 \ldots n\}\|. \tag{5}$$

To determine h, errors obtained after initial training phase should be used.

3.2 Iterative Least Median of Squares

LMedS Estimator. The least median of squares estimator (LMedS) was originally proposed by Rousseeuw [22] but it was informally used even earlier [13]. The LMedS estimator acts on the squared residuals, replacing sum by the robust median, so it can be defined as follows:

$$\hat{\theta} = \arg\min_{i} \text{med}\ {r_i}^2. \tag{6}$$

Iterative LMedS. In the domain of robust neural network learning algorithms, the LMedS error criterion was proposed by El-Melegy in [5], where simulated annealing was employed to minimize the median error. The LMedS performance is defined as:

$$E_{med} = \text{med}\ {r_i}^2. \tag{7}$$

For the error criterion given by 7, the following additional training procedure was proposed [5, 25]. After an initial training phase, the robust standard deviation (RSD)[21] is calculated as:

$$\sigma_r = 1.4826 * (1 + \frac{5}{(N-p)}) \sqrt{E^*_{med}}, \tag{8}$$

where E^*_{med} is the best achieved LMedS error value (N and p are the size of the training set and the dimension of the input vector). Then all the training patterns associated with residuals exceeding a threshold should be removed from the training set:

$$r_i^2 \geq 2.5 * \sigma_r^2. \tag{9}$$

These steps should be repeated iteratively several times. A detailed explanation of the chosen threshold and methodology can be found in [5, 21, 25].

To train the network with LTA and ILMedS approaches we decided to use non-gradient VSS algorithm [14] to cope with the problem of the performance function non-differentiability.

4 Outlier Reduction

4.1 Instance Selection

Using instance selection, the most of the outliers get removed from the training dataset and the noise in the data is reduced. In general there may by also other reasons for instance selection, as reducing the data size or improving generalization, but these topics are out of scope of this work. A large survey of about 70 different instance selection algorithms for classification tasks can be found in [27]. So far there were very few approaches in the literature to instance selection for regression problems. Moreover, the approaches were verified only on artificial datasets generated especially for the purpose of testing the algorithms. Zhang [31] presented a method to select the input vectors while calculating the output with k-NN. Tolvi [28] presented a genetic algorithm to perform feature and instance selection for linear regression models. In their works Guillen et al. [10] discussed the concept of mutual information used for selection of prototypes in regression problems.

Instance selection for regression problems is a more complex issue for two reasons. First, in classification it is enough to determine the border between two classes, while in regression the values in each point of the data are important. This results in a much weaker data compression that can be achieved in regression tasks. And second, in classification we must only decide if a certain points belong to a given class or not. Thus most of the instance selection algorithms are based on k-NN classification, where the result of the classification determines if the given instance is preserved or rejected. In regression problems, while comparing two instances, we consider the distance between them, according to some (usually Euclidean) distance measure. Thus, the criterion to decide whether a given instance should be rejected is some distance threshold. There are a lot of options of how the threshold can be determined. It can be constant or proportional to the local density of the data. In general the threshold should be determined experimentally, but our experiments showed that in the regENN algorithm [15], the rejection threshold θ can be set to 2-8 standard deviations of the data for a broad range of regression problems. The higher value can be used for a better quality data and the lower for highly contaminated data.The reason for this is that in more contaminated data there are more outliers that should be removed and there is a higher probability that the some of the neighbors of the considered instance are also outliers. While in a better quality data even the points that are far from their neighbors do not necessary require rejection, as they may not contain any wrong values. Using θ proportional to the standard deviation of k nearest neighbors of the instance x_i, instead of proportional to a standard deviation of the entire data allows, as the experiments showed, for obtaining higher compression of the dataset while preserving the same prediction accuracy. We developed the regENN algorithm from the ENN (Edited Nearest Neighbor) algorithm [30] and presented it in [15]. The main idea of the regENN algorithm is to reject instances if their output differs more than θ from a value predicted by the weighted k-NN with $k = 9$, where the weight w_i exponentially decreases with the distance d_i between the given instance and its i-th neighbor x_i. The predicted output y is expressed by the following equation:

$$y = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i} \tag{10}$$

where $w_i = 2^{-0.2d_i}$. As the regression model to predict the output $Y(x_i)$ we use k-NN with $k = 9$ as the Model(T,x_i) we also use k-NN with $k = 9$, although also other methods can be used here, as neural network, regression trees, etc. ($k = 9$ was evaluated experimentally to be a good choice for a broad range of problems [16]).

Algorithm 1. regENN algorithm

Require: $\mathbf{T}$
  $m \leftarrow sizeof(\mathbf{T})$;
  for $i = 1 \ldots m$ **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 $\mathbf{P}$

4.2 Anomaly Detection

Anomaly detection deals with the outliers in a different way than instance selection; it does not reject or keep them but it assigns an anomaly score to each instance. The higher the score, the bigger outlier is the instance. There is a bunch of anomaly detection methods and a survey of them can be found in [2]. For the purpose of this work we modified the k-NN Global Anomaly Score algorithm (k-NN GAS). The k-NN GAS assigns the anomaly scores prior to the network training and then the MLP error function divides the error the network makes on the instance by the instance anomaly score. In this way the more outstanding instances have weaker influence on the network training. The advantage of anomaly detection over instance selection is that we do not have to make a crisp decision about the instance. The k-NN GAS calculates the anomaly score based on the k-NN algorithm. The outlier score of an instance is the average distance between the instance and its k nearest neighbors (again we use k = 9 and Euclidean distance measure). However, for the purpose of labeled data, we had to modify the k-NN GAS, including both distances: in the input space d_x and in the output space d_y. We define the modified anomaly score A_{sc} as:

$$A_{sc} = d_y / d_x \tag{11}$$

5 Experimental Evaluation

5.1 Datasets

We performed the experiments on two groups of regression problems: the real-world datasets and artificial datasets. We used the real-world datasets, which were first standardized so that the mean value of each attribute is zero and the standard deviation is one to make result comparison easier. We started from the original datasets (δ=0 in the tables 5-7) and gradually were adding some random noise to outputs only to the training subsets in the crossvalidation. δ=0.1 represents v=0.5 and f=0.20, δ=0.2: v=1.0 and f=0.25, δ=0.3: v=1.5 and f=0.30, δ=0.4: v=2.0 and f=0.35, δ=0.5: v=2.5 and f=0.40. The noise was added to outputs with random frequency f and amplitude $v(2 - r * r)$, where $0 < r < 1$ is a random number. The artificial datasets (Function A, Function B and Function C) and Building Benchmark were contaminated with so-called Gross Error Model [3, 4, 18, 23] with additive noise: $F = (1-\delta)G + \delta H$, where F denotes the error distribution, $G \sim N(0.0, 10.0)$ models small Gaussian noise, and $H \sim N(0.0, 0.1)$ represents high value outliers. Hence, the probability of outliers is δ. The datasets are available from [32].

Function A. The 1-D function to be approximated was proposed by Liano in [18] and used to test many robust learning algorithms [3–5, 19, 26]. It is defined as:

$$y = |x|^{-2/3}. \tag{12}$$

A training set was prepared by sampling independent variable in the range $[-2, 2]$ with a step 0.01.

Function B. The second 1-D function was previously used in [3, 4] and defined as:

$$y = \frac{\sin(x)}{x}. \tag{13}$$

For a training set, the independent variable was sampled in the range $[-7.5, 7.5]$ with a step of 0.1.

Function C. Another function was a two-dimensional spiral defined as:

$$\begin{cases} x = \sin y \\ z = \cos y \end{cases} \tag{14}$$

Training data were generated by sampling the dependent variable y in the range $[0, \pi]$ with a step $\pi/100$. The network was trained to model y as a function of x and z (for the given range it is a function).

Building. The first real-world training task was taken from the PROBEN 1 benchmark collection [20]. The task was to predict building energy consumption based on 14 input variables, such as the date, time, and weather conditions. Following [1], we trained a network on the first 3156 observations to predict dependent variable over the next 1052 time steps of the test set.

Table 1. MSE on training subset for Function A, 10 hidden neurons, 12 training epochs

δ	0.0	0.1	0.2	0.3	0.4	0.5
MSE	0.0038±0.002	0.11±0.006	0.44±0.008	0.49±0.08	0.59±0.09	0.81±0.09
ILMedS	0.0045±0.001	0.0057±0.003	0.0076±0.003	0.0059±0.002	0.015±0.005	0.021±0.01
LTA	0.0065±0.001	0.0041±0.001	0.011±0.005	0.0046±0.002	0.0063±0.002	0.0073±0.002
ENN-MSE	0.0039±0.002	0.0055±0.002	0.0077±0.003	0.013±0.030	0.014±0.033	0.018±0.028
ENN-ILMedS	0.0048±0.001	0.0061±0.002	0.0070±0.002	0.0071±0.002	0.0092±0.003	0.014±0.023
ENN-LTA	0.0039±0.001	0.0039±0.001	0.0048±0.002	0.0055±0.002	0.0059±0.002	0.0067±0.002
GAS-MSE	0.0033±0.002	0.0044±0.002	0.0048±0.002	0.0061±0.002	0.0087±0.002	0.017±0.004
GAS-ILMedS	0.0037±0.001	0.0042±0.002	0.0056±0.002	0.0088±0.002	0.021±0.005	0.067±0.019
GAS-LTA	0.0021±0.001	0.0020±0.001	0.0023±0.001	0.0027±0.001	0.0035±0.001	0.0068±0.002

Table 2. MSE on training subset for Function B, 10 hidden neurons, 12 training epochs

δ	0.0	0.1	0.2	0.3	0.4	0.5
MSE	0.0044±0.002	0.67±0.12	0.45±0.05	0.45±0.03	1.90±0.09	4.66±0.10
ILMedS	0.0045±0.002	0.046±0.017	0.024±0.011	0.056±0.035	0.11±0.03	0.15±0.09
LTA	0.0072±0.002	0.0056±0.002	0.0091±0.002	0.010±0.005	0.021±0.007	0.15±0.04
ENN-MSE	0.0034±0.001	0.0038±0.001	0.0055±0.002	0.0053±0.002	0.0081±0.002	0.027±0.007
ENN-ILMedS	0.0030±0.001	0.0040±0.003	0.0049±0.003	0.0056±0.002	0.0066±0.003	0.018±0.005
ENN-LTA	0.0038±0.001	0.0036±0.001	0.0040±0.002	0.0048±0.002	0.0076±0.003	0.015±0.004
GAS-MSE	0.0031±0.001	0.0049±0.002	0.0068±0.003	0.011±0.003	0.023±0.006	0.082±0.031
GAS-ILMedS	0.0042±0.001	0.0051±0.002	0.0067±0.003	0.013±0.003	0.034±0.011	0.18±0.06
GAS-LTA	0.0040±0.002	0.0043±0.002	0.0045±0.002	0.0045±0.002	0.0062±0.003	0.021±0.006

Table 3. MSE on training subset for Function C, 10 hidden neurons, 12 training epochs

δ	0.0	0.1	0.2	0.3	0.4	0.5
MSE	0.0025±0.001	0.19±0.02	0.60±0.05	1.62±1.77	2.58±0.48	4.16±0.30
ILMedS	0.0021±0.001	0.015±0.012	0.081±0.039	1.74±0.94	1.14±0.78	1.62±1.77
LTA	0.0008±0.001	0.0041±0.002	0.0045±0.002	0.014±0.009	0.011±0.004	0.057±0.031
ENN-MSE	0.0025±0.001	0.0041±0.002	0.0061±0.002	0.014±0.005	0.023±0.008	0.044±0.012
ENN-ILMedS	0.0020±0.001	0.0043±0.002	0.0087±0.003	0.017±0.006	0.022±0.005	0.039±0.01
ENN-LTA	0.0008±0.001	0.0018±0.001	0.0044±0.002	0.010±0.003	0.021±0.002	0.039±0.002
GAS-MSE	0.0022±0.001	0.0039±0.002	0.0056±0.008	0.015±0.004	0.044±0.009	0.1415±0.09
GAS-ILMedS	0.0024±0.001	0.0065±0.003	0.0077±0.003	0.013±0.003	0.092±0.035	0.34±0.01
GAS-LTA	0.0014±0.001	0.0035±0.001	0.0048±0.005	0.0042±0.002	0.0046±0.002	0.054±0.018

Concrete Compression Strength. There are 1030 instances with 7 input attributes in the dataset reflecting the amount of particular substances in the concrete mixture, such as cement, slag, water, etc. [29]. The task is to predict the concrete compressive strength. There are 1030 instances in the database.

Crime and Communities. There are 318 instances with originally 120 input attributes in the data set, describing various social, economical and criminal factors [29].

Table 4. MSE on training subset for Building dataset, 10 hidden neurons, 12 training epochs

δ	0.0	0.1	0.2	0.3	0.4	0.5
MSE	0.0018±0.0003	1.013±0.030	3.85±0.06	8.29±0.13	15.6±0.2	24.3±2.2
ILMeds	0.0017±0.0003	0.036±0.019	0.16±0.06	0.25±0.13	0.41±0.18	15.2±1.5
LTA	0.0020±0.0004	0.0032±0.0006	0.0048±0.001	0.013±0.003	0.026±0.004	2.15±3.9
ENN-MSE	0.0018±0.0003	0.0039±0.0006	0.0060±0.012	0.018±0.004	0.034±0.006	0.24±0.05
ENN-ILMedS	0.0017±0.0003	0.0035±0.0006	0.0056±0.001	0.015±0.002	0.040±0.008	0.17±0.04
ENN-LTA	0.0020±0.0004	0.0035±0.0005	0.0081±0.002	0.014±0.002	0.029±0.005	0.21±0.05

However, after preliminary feature selection we used only 7 attributes. The value to predict is per capita violent crime.

SteelC14. The dataset contains 2384 instances with 18 input attributes. The task is to predict the amount of carbon that must be added in the steel-making process, given various chemical and physical properties of the liquid steel in the furnace.

5.2 Experimental Setup

We implemented the algorithms in C#. The source code can be downloaded from the SVN repository at [32]. The whole process in different configurations was run in 10-fold crossvalidation loops. To be able to compare the results, we always measure and report in the tables 1-7 the MSE error on the test sets, no matter which error function was used for the network training. Also the MLP architecture was constant (the same for each training method) for a given dataset (the numbers of hidden neurons are given in the result tables). We run the tests on several forms of the datasets: the original datasets and the datasets with various amount of random noise (see section 5.1) added to the output variables. However, the noise was added only to the training data, while the test data were left unchanged. That allowed us to determine how the methods can deal with various noise levels.

5.3 Results

The results in the tables show MSE on the test subsets (always MSE on the test subset is compared for any training method and any error function used during the training). Analyzing results of the experiments, one may notice that the traditional method, minimizing MSE criterion perform well only for clean datasets without outliers. When the data contains outlying patterns, the method breaks down. More interesting phenomenon is that even for clean training data, different modified algorithms always obtained better results (e.g. GAS methods in Table 1, or ENN and GAS methods in Table 2).

For contaminated training sets, all the enhanced algorithms performed better than the traditional one. Only pure ILMedS method for several datasets (Tables 5, 6, 7) obtained lager errors than the MSE. In general, the best performance was achieved for hybrid algorithms combining ILMedS and LTA with ENN, or GAS approaches.

Table 5. MSE on training subsets for Concrete dataset, 6 hidden neurons, 12 training epochs

δ	0.0	0.1	0.2	0.3	0.4	0.5
MSE	0.79±0.25	0.84±0.21	1.01±0.19	1.47±0.22	2.26±0.19	3.80±0.33
ILMedS	0.91±0.25	1.05±0.36	1.14±0.34	2.01±0.46	2.80±0.85	4.01±1.30
LTA	1.06±0.33	0.96±0.39	0.94±0.35	0.94±0.32	1.07±0.29	1.60±0.35
ENN-MSE	0.79±0.25	0.82±0.21	0.89±0.20	1.02±0.14	1.05±0.15	1.15±0.18
ENN-ILMedS	0.85±0.09	0.86±0.11	0.89±0.11	0.88±0.15	0.98±0.21	1.09±0.17
ENN-LTA	1.05±0.20	0.95±0.16	0.94±0.09	1.00±0.18	1.13±0.25	1.21±0.18
GAS-MSE	0.78±0.32	0.87±0.36	0.94±0.29	1.13±0.28	1.41±0.25	2.16±0.39
GAS-ILMedS	0.76±0.26	0.97±0.38	1.08±0.30	1.21±0.43	1.64±0.34	2.80±0.81
GAS-LTA	1.10±0.30	1.09±0.44	1.03±0.40	0.96±0.33	1.00±0.34	1.05±0.33

Table 6. MSE on training subsets for Crime dataset, 5 hidden neurons, 12 training epochs

δ	0.0	0.1	0.2	0.3	0.4	0.5
MSE	0.34±0.07	0.37±0.10	0.56±0.17	1.17±0.41	2.23±0.42	3.31±0.71
ILMedS	0.37±0.12	0.43±0.10	0.63±0.13	1.37±0.44	2.70±1.11	4.09±1.33
LTA	0.39±0.11	0.39±0.10	0.48±0.13	0.56±0.18	0.88±0.24	1.96±0.92
ENN-MSE	0.34±0.07	0.34±0.10	0.40±0.07	0.61±0.19	0.70±0.49	1.58±1.38
ENN-ILMedS	0.37±0.11	0.36±0.09	0.46±0.15	0.53±0.11	0.69±0.21	0.77±0.35
ENN-LTA	0.38±0.13	0.38±0.12	0.45±0.12	0.54±0.14	0.64±0.40	0.81±0.30
GAS-MSE	0.34±0.09	0.39±0.12	0.47±0.13	0.61±0.18	1.22±0.47	2.43±0.83
GAS-ILMedS	0.34±0.10	0.37±0.11	0.47±0.10	0.86±0.62	1.51±0.72	2.99±1.23
GAS-LTA	0.37±0.11	0.39±0.10	0.46±0.12	0.47±0.14	0.61±0.16	1.08±0.63

Table 7. MSE on training subsets for SteelC14, 5 hidden neurons, 12 training epochs

δ	0.0	0.1	0.2	0.3	0.4	0.5
MSE	0.071±0.018	0.10±0.03	0.27±0.02	0.70±0.08	1.61±0.12	3.14±0.16
ILMedS	0.082±0.035	0.17±0.04	0.55±0.14	1.07±0.41	2.06±0.86	2.58±1.05
LTA	0.071±0.045	0.069±0.038	0.093±0.034	0.11±0.05	0.15±0.04	0.27±0.07
ENN-MSE	0.069±0.016	0.072±0.031	0.92±0.10	0.10±0.04	0.14±0.04	0.22±0.06
ENN-ILMedS	0.070±0.014	0.098±0.023	0.21±0.03	0.55±0.02	0.78±0.18	1.13±0.56
ENN-LTA	0.068±0.015	0.111±0.029	0.24±0.03	0.60±0.02	0.12±0.03	0.21±0.07
GAS-MSE	0.073±0.045	0.084±0.021	0.121±0.010	0.27±0.05	0.69±0.08	1.30±0.32
GAS-ILMedS	0.071±0.034	0.110±0.050	0.221±0.140	0.60±0.25	0.68±0.20	1.31±0.41
GAS-LTA	0.073±0.054	0.074±0.036	0.074±0.043	0.078±0.04	0.09±0.04	0.15±0.05

6 Conclusions

We described briefly some modifications of learning methods designed to deal with the problem of noisy data for regression tasks. It is clearly evident that all the presented approaches can be considered as more reliable than the traditional learning algorithms minimizing the MSE criterion. This is particularly important when the quality of training data is unknown. Even for clean training patterns some of the modified methods performed better than the MSE. For different testing problems and different amounts of outliers the observed performances varied between tested methods. However, in most cases, especially for the noisy data, ENN with LTA performed best. The future efforts can be then directed at defining and choosing optimal algorithms for given conditions (types of problems and quantities of outlying points).

References

1. Beliakov, G., Kelarev, A., Yearwood, J.: Derivative-free optimization and neural networks for robust regression. Optimization 61(12), 1467–1490 (2012)
2. Ben-Gal, I.: Outlier detection. Kluwer Academic Publishers (2005)
3. Chen, D., Jain, R.: A robust backpropagation learning algorithm for function approximation. IEEE Transactions on Neural Networks 5(3), 467–479 (1994)
4. Chuang, C.C., Su, S.F., Hsiao, C.C.: The annealing robust backpropagation (arbp) learning algorithm. IEEE Transactions on Neural Networks 11(5), 1067–1077 (2000)
5. El-Melegy, M.T., Essai, M.H., Ali, A.A.: Robust training of artificial feedforward neural networks. In: Hassanien, A.-E., Abraham, A., Vasilakos, A.V., Pedrycz, W. (eds.) Foundations of Computational, Intelligence Volume 1. SCI, vol. 201, pp. 217–242. Springer, Heidelberg (2009)
6. El-Melegy, M.: Random sampler m-estimator algorithm for robust function approximation via feed-forward neural networks. In: The 2011 International Joint Conference on Neural Networks (IJCNN), pp. 3134–3140 (2011)
7. El-Melegy, M.: Ransac algorithm with sequential probability ratio test for robust training of feed-forward neural networks. In: The 2011 International Joint Conference on Neural Networks (IJCNN), pp. 3256–3263 (2011)
8. 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), 1074–1085 (2013)
9. Golak, S., Burchart-Korol, D., Czaplicka-Kolarz, K., Wieczorek, T.: Application of neural network for the prediction of eco-efficiency. In: Liu, D., Zhang, H., Polycarpou, M., Alippi, C., He, H. (eds.) ISNN 2011, Part III. LNCS, vol. 6677, pp. 380–387. Springer, Heidelberg (2011)
10. Guillen, A.: Applying mutual information for prototype or instance selection in regression problems. In: ESANN 2009 (2009)
11. Hampel, F.R., Ronchetti, E.M., Rousseeuw, P.J., Stahel, W.A.: Robust Statistics: The Approach Based on Influence Functions (Wiley Series in Probability and Statistics), revised edn. Wiley-Interscience, New York (2005)
12. Hart, P.: The condensed nearest neighbor rule (corresp.). IEEE Transactions on Information Theory 14(3), 515–516 (1968)
13. Huber, P.J.: Robust Statistics. Wiley Series in Probability and Statistics. Wiley-Interscience (1981)

14. Kordos, M., Duch, W.: Variable Step Search Algorithm for Feedforward Networks. Neurocomputing 71(13-15), 2470–2480 (2008)
15. Kordos, M., Białka, S., Blachnik, M.: Instance selection in logical rule extraction for regression problems. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013, Part II. LNCS, vol. 7895, pp. 167–175. Springer, Heidelberg (2013)
16. Kordos, M., Blachnik, M., Strzempa, D.: Do We Need Whatever More Than k-NN? In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010, Part I. LNCS (LNAI), vol. 6113, pp. 414–421. Springer, Heidelberg (2010)
17. Kordos, M., Rusiecki, A.: Improving MLP Neural Network Performance by Noise Reduction. In: Dediu, A.-H., Martín-Vide, C., Truthe, B., Vega-Rodríguez, M.A. (eds.) TPNC 2013. LNCS, vol. 8273, pp. 133–144. Springer, Heidelberg (2013)
18. Liano, K.: Robust error measure for supervised neural network learning with outliers. IEEE Transactions on Neural Networks 7(1), 246–250 (1996)
19. Pernia-Espinoza, A.V., Ordieres-Mere, J.B., de Pison, F.J.M., Gonzalez-Marcos, A.: Tao-robust backpropagation learning algorithm. Neural Networks 18(2), 191–204 (2005)
20. Prechelt, L.: Proben1 – a set of neural network benchmark problems and benchmarking rules. Tech. rep. (1994)
21. Rousseeuw, P.J., Leroy, A.M.: Robust Regression and Outlier Detection. John Wiley & Sons, Inc., New York (1987)
22. Rousseeuw, P.J.: Least median of squares regression. Journal of the American Statistical Association 79(388), 871–880 (1984)
23. Rusiecki, A.: Robust LTS backpropagation learning algorithm. In: Sandoval, F., Prieto, A., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 102–109. Springer, Heidelberg (2007)
24. Rusiecki, A.: Robust MCD-based backpropagation learning algorithm. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 154–163. Springer, Heidelberg (2008)
25. Rusiecki, A.: Robust learning algorithm based on iterative least median of squares. Neural Processing Letters 36(2), 145–160 (2012)
26. Rusiecki, A.: Robust learning algorithm based on LTA estimator. Neurocomputing 120, 624–632 (2013)
27. 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, 417–435 (2012)
28. Tolvi, J.: Genetic algorithms for outlier detection and variable selection in linear regression models. Soft Computing 8, 527–533 (2004)
29. Merz, C., Murphy, P.: Uci repository of machine learning databases (2013), `http://www.ics.uci.edu/mlearn/MLRepository.html`
30. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man and Cybernetics SMC-2(3), 408–421 (1972)
31. Zhang, J.: Intelligent selection of instances for prediction functions in lazy learning algorithms. Artifcial Intelligence Review 11, 175–191 (1997)
32. Source code and datasets used in the paper, `https://code.google.com/p/mlp2013/`