Neural Process Lett (2012) 36:145–160
DOI 10.1007/s11063-012-9227-z

Robust Learning Algorithm Based on Iterative Least Median of Squares

Andrzej Rusiecki

Published online: 15 May 2012

Abstract Outliers and gross errors in training data sets can seriously deteriorate the performance of traditional supervised feedforward neural networks learning algorithms. This is why several learning methods, to some extent robust to outliers, have been proposed. In this paper we present a new robust learning algorithm based on the iterative Least Median of Squares, that outperforms some existing solutions in its accuracy or speed. We demonstrate how to minimise new non-differentiable performance function by a deterministic approximate method. Results of simulations and comparison with other learning methods are demonstrated. Improved robustness of our novel algorithm, for data sets with varying degrees of outliers, is shown.

Keywords Feedforward neural networks · Robust learning algorithms · Outliers · Robust statistics

1 Introduction

Feedforward artificial neural networks (FNN) have been successively applied in areas such as function approximation, pattern recognition or signal and image processing. Because the FNNs are universal approximators [9,10], they can potentially be used in any type of problems that require modelling of unknown input–output dependencies. Such networks build their models based on training sets consisting of exemplary input–output patterns. The main advantage of such approach is its simplicity, since any prior knowledge about modelled system is not required. These networks are usually trained to minimise an error function defined to measure the distance between the current and desired output. During the training process, FNNs try to fit the training data as close as possible. Unfortunately, the performance of this type of learning scheme relies strongly on the quality of training data [8,11,15]. When the data are corrupted with large noise or outliers the network is trained on erroneous examples

A. Rusiecki (✉)
Wroclaw University of Technology, Wroclaw, Poland
e-mail: andrzej.rusiecki@pwr.wroc.pl

and tries to model a system different from the desired one. This is because the most popular backpropagation (BP) learning algorithm and many of its variants use the mean squared error (MSE) function. This strategy, based on the least mean squares method, is optimal only for the clean data or data with normal error distribution.

Outliers may be defined as observations deviating strongly from the majority of the data. Unfortunately, in routine data, the quantity of outliers can range from 1 to 10 % [8], or in certain cases even more. They may be caused by measurement errors, human mistakes such as errors in copying or wrong decimal points, long-tailed noise resulting in different sample distribution, measurements of members of wrong populations, rounding errors and many other reasons.

When we deal with multidimensional data set, finding even one outlying observation involves computationally expensive methods. In the case when more outliers exist, the situation becomes obviously much more complicated.

In this paper, we present a new learning algorithm that is robust to various degrees of outlying data in training sets. The novel algorithm takes advantage of the idea of the least median of squares estimator. It is applied iteratively to remove outliers from the training data, but it provides also satisfactory performance when the network is trained on the clean data set.

2 Network Training with Outliers

The feedforward networks learning algorithms, that are based on the minimisation of some kind of criterion function, use backpropagation to calculate the performance gradient with respect to network weights (and biases which may be also considered as additional weights). To introduce network performance function, let us consider, without loss of generality, a simple three layer feedforward neural network with one hidden layer. We assume that the training set consists of N pairs:
$\{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \ldots, (\mathbf{x}_N, \mathbf{t}_N)\}$, where $\mathbf{x}_i \in R^p$ denotes the p-dimensional ith input vector and $\mathbf{t}_i \in R^q$ the corresponding q-dimensional network target. For the given input vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$, the output of the jth of l neurons of the hidden layer may be calculated as:

$$z_{ij} = f_1\left(\sum_{k=1}^{p} w_{jk}x_{ik} - b_j\right) = f_1(inp_{ij}), \quad \text{for } j = 1, 2, \ldots, l, \tag{1}$$

where $f_1(\cdot)$ is the activation function of the hidden layer, inp_{ij} is the sum of its weighted inputs, w_{jk} is the weight between the kth net input and jth neuron, and b_j is the bias of the jth neuron. For such network its output $\mathbf{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iq})^T$ is given as:

$$y_{iv} = f_2\left(\sum_{j=1}^{l} w'_{vj}z_{ij} - b'_v\right) = f_2(inp_{iv}), \quad \text{for } v = 1, 2, \ldots, q. \tag{2}$$

Here $f_2(\cdot)$ denotes the output layer activation function, w'_{vj} is the weight between the vth neuron of the output layer and the jth neuron of the hidden layer, and b'_v is the bias of the vth neuron of the output layer. When f_1 and f_2 are similar, these equations can be simplified, however for the function approximation or regression task, the most common approach is to use the sigmoid activation function in the hidden layers and linear activation in its output.

For the residuals r_i written as:

$$r_i = \sum_{v=1}^{q} |(y_{iv} - t_{iv})|, \tag{3}$$

the performance function may be defined as:

$$E = \frac{1}{N} \sum_{i=1}^{N} \rho\,(r_i), \tag{4}$$

where $\rho(r_i)$ is a symmetric and continuous loss function [8], r_i is an error for the i-th training pattern (3), and N is the number of elements in the training set. The most popular loss function is of quadratic form:

$$\rho(r_i) = \frac{{r_i}^2}{2}. \tag{5}$$

For the quadratic loss function we obtain the minimised error equal to the MSE:

$$E_{mse} = \frac{1}{N} \sum_{i=1}^{N} {r_i}^2. \tag{6}$$

The influence function [8,14] was introduced to measure the impact of data errors to the training process. It may be defined as a derivative of the loss function with respect to residuals:

$$\psi(r_i) = \frac{\partial \rho(r_i)}{\partial r_i}. \tag{7}$$

If we assume the MSE performance function, then the influence function becomes linear:

$$\psi(r_i) = r_i, \tag{8}$$

which means the larger the error, the more it affects the training process. Since large errors are often caused by outliers, this phenomenon seems to be very dangerous. This is why various robust learning algorithms based on robust estimators have been proposed [1,2,14,20].

3 Robust Learning Algorithms

In the field of robust statistics [8,11] many methods to deal with the problem of outliers have been proposed. They are designed to act properly when the true underlying model deviates from the assumptions, such as normal error distribution. There are robust methods that detect and remove outlying data before the model is built, but more of them, including robust estimators, should be efficient and reliable even if outliers appear. Simultaneously, they should perform well for the observations that are very close to the assumed model.

The simplest idea to make the traditional neural network learning algorithm more robust to outliers is to replace the quadratic error with another symmetric and continuous loss function, resulting in the nonlinear influence function. Such nonlinearity should reduce the influence of large errors. Robust loss functions can be based on the robust estimators with proved ability to tolerate different amounts of outlying data. Replacing the MSE performance function with a new robust function results in robust learning method with the reduced impact of outliers.

Several such algorithms destined to the FNNs with sigmoid activation functions, have been proposed.

One of the first robust learning algorithms, the LMLS (Least Mean Log Squares) method, was introduced by Liano [14]. He proposed the logistic error function, derived from the assumption of the errors generated with the Cauchy distribution. This contribution was considered as referential by other authors who tried to construct more efficient functions. The idea of so-called M-estimators [8] was applied by Chen and Jain [1] in using the Hampel's hyperbolic tangent as a new error criterion. For this performance function additional scale estimator β, defining the size of residuals suspected to be outliers, was also introduced. In [2] Chunag and Su applied the annealing scheme to decrease the value of β with the progress of training. There were also approaches with performance functions based on the tau-estimators [16] and the LTS (Least Trimmed Squares) estimator, while initial data analysis with the MCD (Minimum Covariance Determinant) estimator was proposed in [20]. El-Melegy et al. in [6] have presented the Simulated Annealing for Least Median of Squares (SA-LMedS) algorithm, applying the simulated annealing technique to minimise performance measured by the median of squared residuals. Some efforts to make the learning methods of radial basis function networks more robust, following the approaches for the sigmoid networks, have been also made [3,4]. The most recent robust learning methods to be mentioned are robust co-training based on the canonical correlation analysis proposed by Sun and Jin [22], and robust adaptive learning using linear matrix inequality techniques [13].

In this paper we present a new robust learning algorithm based on the iterated Least Median of Squares (LMedS) estimator. The novel approach is much more effective and significantly faster than the SA-LMedS method [6]. It achieves also better resistance to erroneous training data. To make the training process more robust, we modify not only the performance function but also remove iteratively data suspected to be outliers. Moreover, we propose an approximate method to minimise the LMedS error criterion.

4 Least Median of Squares Estimator

The least median of squares estimator (LMedS) is known in the robust statistics literature as having high breakdown point. It was originally proposed by Rousseeuw [18] but its informal use can be dated even earlier [11]. The breakdown point [8] is the smallest percentage ϵ^* of contaminated data that can cause the estimator to take on aberrant values. In the case of the least squares method it is obviously $\epsilon^* = 0$. What may be surprising, the same breakdown point $\epsilon^* = 0$ is achieved by the M-estimators and least absolute values (L_1 norm). As an alternative the LMedS can be used. It can theoretically tolerate up to 50 % outliers in the processing data having $\epsilon^* = 0.5$ [18]. Unfortunately it has at the same time very low efficiency, this is why other robust estimators are more popular.

Unlike the M-estimators, the LMedS estimator acts on the squared residuals. The sum is replaced by the robust median resulting in minimisation problem of the form:

$$\min_i \operatorname{med} {r_i}^2. \tag{9}$$

In [6] the authors described a robust learning method based on the LMedS error criterion (SA-LMedS algorithm). The LMedS performance function can be written as:

$$E_{med} = \operatorname{med} {r_i}^2. \tag{10}$$

For such performance function the main problem is that it cannot be minimised by any gradient algorithm. This is why the Simulated Annealing (SA) technique was proposed. It possesses, however, many disadvantages. The first one is its stochastic nature. Because the SA uses random search to find a solution, it works relatively slow. Though it may lead the training process into the region close to global minimum, it is less efficient in finding its exact value. Another problem is the proper setting of algorithm parameters. To train the network with LMedS performance function, the authors in [6] used different sets of parameters for each tested network and each testing case. The methodology was to perform manually several tuning trials before obtaining the best values for the algorithm parameters, because improperly chosen parameters would significantly decrease the method performance. If the network was trained without preliminary knowledge of the data (without possibility to be tested on the clean data set), it would be almost impossible to judge the quality of different values of parameters. The last problem was poor efficiency of the LMedS estimator for the Gaussian data set. The authors suggested then to train the network once again with traditional backpropagation (minimizing the MSE), after applying the SA-LMedS algorithm.

It is clearly evident that the SA algorithm is unfortunately very slow and inaccurate in comparison to gradient based learning methods (even to simple steepest descent algorithm). Ideally we'd like to have some kind of deterministic algorithm to minimise the LMedS network error function. It is probably impossible, due to the fact that a formula to reduce the LMedS estimator to weighted least problem is not known [6]. Moreover, from the technical point of view, the LMedS error function is non-differentiable, so its gradient value cannot be calculated. Our idea is then to estimate the LMedS gradient or use some heuristics that could lead the training process in the right direction. The method not only employs the idea of replacing the error criterion by a new function, based on more robust estimator, but also introduces a new learning procedure to overcome the problem of poor Gaussian efficiency (the LMedS estimator converges like $n^{-1/3}$ [18,19], and when it is used as the error function it requires retraining the network with the MSE criterion [6]).

5 Robust Learning Algorithm with LMedS Criterion

As previous research efforts demonstrated, it is possible to train the FNNs with median neuron input function with gradient-based algorithms [21]. In these networks, simple summation in the neuron input is replaced by the median input function, which also causes non-differentiability of the network error function. In this case, the gradient was estimated as depending on regular summation input function. Such estimation was good enough to make the training process effective in minimising the error function.

In the case of the LMedS error function, we may try then to use similar strategy. The situation is, of course, more complicated, or worse, in the terms of gradient estimation accuracy. Minimising the LMedS error based directly on the gradient calculated for the MSE error function should not work. Our approach is to try to use a method based on the well-known resilient backpropagation (RBP) algorithm [17].

5.1 Resilient Backpropagation for the LMedS Criterion

In the RBP algorithm, similarly to other gradient-based learning methods, the backpropagation strategy is used to calculate derivatives of the performance function with respect to the network weights. However, unlike the other methods, the RBP uses only the sign of the gradient to determine the direction of network update [17]. Several improvements to the RBP

algorithm have been proposed [12] but we decided to use its original version, mainly due to its simplicity. The RBP learning algorithm was developed to eliminate the effect of slow convergence for the very low gradient magnitude caused by the flat regions of sigmoid activation functions. This is why it seems to be more suitable to the LMedS error function, because proper estimation of the sign of the gradient is more likely than proper estimation of its exact value. The idea is then to calculate derivatives, as for the MSE performance function given by Eq 6, and use them to minimise the LMedS error given by (10). Each step of the learning algorithm is divided into calculating gradient sign as in the RBP algorithm and evaluating the LMedS-based performance. The weights update can be written as follows:

$$\Delta \mathbf{w}(k) = \boldsymbol{\delta}(k) * \text{sign}(\nabla \mathbf{E}_{\text{mse}}), \tag{11}$$

where $\nabla \mathbf{E}_{\text{mse}}$ denotes gradient of the E_{mse} performance, k is the current epoch, and elements of $\boldsymbol{\delta}$ for each network weight separately. Starting from the initial value $\boldsymbol{\delta}(0)$ at each iteration, these elements are modified based on corresponding gradient components. If the derivative of the performance function with respect to the weight i has the same sign for two iterations, the corresponding element δ_i is increased as:

$$\delta_i(k+1) = a_1 \times \delta_i(k), \tag{12}$$

where a_1 is an increasing factor. Whenever the gradient component changes its sign, the element δ_i is then decreased as:

$$\delta_i(k+1) = \delta_i(k)/a_2, \tag{13}$$

where a_2 is a decreasing factor. It helps in avoiding weights oscillations, and from the other side, speeds up the training process in the flat regions of the performance function. This method is considered to be not very sensitive to the parameter settings, so we decided to use the values originally proposed in [17].

The steps of the approximate RBP algorithm for the LMedS error criteria can be then written as follows:

1. Use backpropagation to calculate derivatives of the MSE performance function (6).
2. Update the network weights (11).
3. Modify elements of $\boldsymbol{\delta}$ according to (12, 13).
4. Calculate the network LMedS performance E_{med} (10).
5. If the LMedS performance is minimised to the assumed goal, or the number of epochs exceeds the maximum number of epochs, stop training. Otherwise go to step one.

As our experiments have shown, such learning method often does not provide satisfactory solutions. Moreover, this algorithm is based on the gradient sign calculated for the MSE function, which makes it only approximate in finding proper direction for the LMedS function minimisation. This is why we propose to use it only as a part of the main algorithm described in the next subsection.

5.2 Iterative LMedS Algorithm

What we propose next is then to minimise the LMedS error iteratively, in each step reducing the number of training data by rejecting those suspected to be outliers. After initial training, we calculate robust standard deviation (RSD)[19]:

$$\sigma_r = 1.4826 \times \left(1 + \frac{5}{(N-p)}\right) \sqrt{E^*_{\text{med}}}, \tag{14}$$

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, as defined in Sect. 2), and remove from the training set all patterns associated with residuals exceeding a threshold based on the RSD:

$$r_i^2 \geq 2.5 \times \sigma_r^2. \tag{15}$$

Then the network is retrained and new value of RSD is calculated. This procedure should be repeated several times in order to achieve the assumed level of the LMedS performance. The constant 1.4826 is chosen to provide better efficiency for the clean data with Gaussian noise, and the factor $\left(1 + \frac{5}{(N-p)}\right)$ is to compensate the effect of small sample size [6]. The more detailed explanation of the meaning of these numbers can be found in [19].

The RSD calculated after each network training, determines the border between outliers and majority of the clean data. In the next step the data set is reduced, so the training process can be more accurate. After a few iterations the LMedS performance on the current data is usually minimised to the assumed value.

The main algorithm can be written in the following steps:

1. Train the network on the whole data set using approximate RBP algorithm to minimise E_{med} (10).
2. If the criterion function E_{med} is below minimal value, stop training.
3. Remove outliers from the current data set using the best criterion function value E^*_{med} and current RSD (14 and 15).
4. Retrain the network on the new data without outliers, minimising E_{med}.
5. If the criterion function E_{med} is below minimal value or the maximum number of iterations was reached, stop training. Otherwise go to step three.

Unlike the SA-LMedS algorithm this method does not require a set of user-supplied parameters. The only values to be chosen are maximum number of algorithm iterations and the goal of the error criterion. Also no additional training by the traditional MSE-based algorithm is required to achieve acceptable performance for the clean data set without outliers. Rejecting data suspected to be outliers in each algorithm iteration provides that the network performance, even for high percentage of outlying data, is relatively stable. Our experiments suggest that the method is not very sensitive to parameter changes.

6 Simulation Results

To test our novel algorithm and compare its performance with other methods, we decided to follow the experiments described in [6] as close as possible. Such approach should allow to compare our results with the results obtained for the existing robust learning algorithms. We present here the effects of training by regular algorithm minimising the MSE function, robust LMLS algorithm, and our robust ILMedS method. The LMLS algorithm is considered as referential in many publications about robust network training [1,2,6,14,16,20]. We do not include here results obtained for the SA-LMedS algorithm because of two reasons. First of all, the detailed discussion of the SA-LMedS performance for the considered testing tasks may be found in [6] and the results are indeed similar to what we obtained in our experiments. The second reason is that the SA-LMedS method is very sensitive to parameter settings, so each run of the algorithm should be preceded by the parameter tuning, which makes statistical comparison much more difficult.

6.1 Error Models

To model gross errors in the training data set we decided to use two models well-known from the robust learning literature. The first one was so-called Gross Error Model (GEM) [1,14], the second one was based on substituting data points by the background noise [6,16]. The estimated percentage of outliers was varied from $\delta = 0$ to $\delta = 50\,\%$ by changing the probability of the occurrence of outliers in the range [0, 0.5] with a step 0.1.
Gross Error Model In this model the clean data are corrupted with additive noise $F = (1 - \delta)G + \delta H$, where F is the error distribution, G models small Gaussian noise, and H - outliers. $G \sim N(0.0, 0.1)$ and $H \sim N(0.0, 2.0)$ occur with probabilities $1 - \delta$ and δ respectively.
Background Noise Randomly selected data points are substituted with probability δ with a background noise uniformly distributed in the range $[-2, 2]$.

6.2 One-Dimensional Function Approximation

Function A To test our algorithm on the 1-D function approximation task, we decided to use the function proposed first by Liano in [14] and employed to test many other robust algorithms [1,2,6,20]. This function is given by the equation:

$$y = |x|^{-2/3}. \tag{16}$$

The data points were generated by sampling the independent variable in the range $[-2, 2]$ with a step 0.01, and calculating the dependent variable by (16). Such clean patterns were then contaminated according to one of the error models. The approximated function and exemplary training data were shown in the Fig. 1. The network architecture was a FFN with one input, five hidden neurons with sigmoid transfer functions and one linear output neuron. The MSE and LMLS criterion were minimised by the conjugate–gradient algorithm [7].

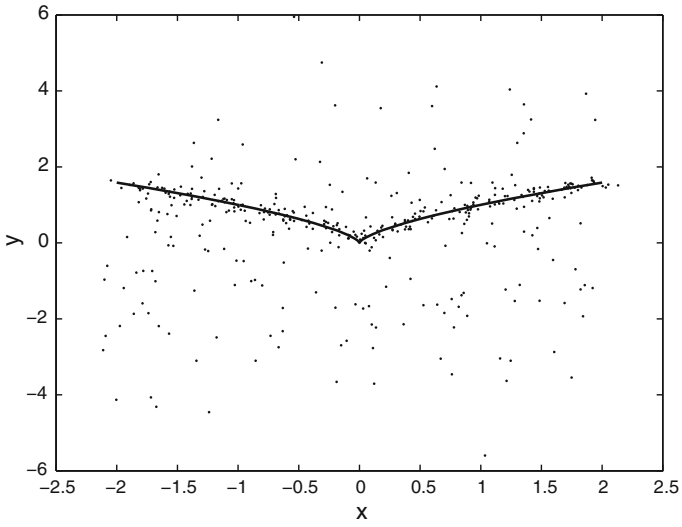

Fig. 1 Training data for the 1-D function approximation (function A) with Gross Error Model, $\delta = 0.5$ and the ideal function *solid line*

Table 1 The mean MSE for the networks trained to approximate function A (data with gross errors)

	Data with gross errors $\delta = 0.0$		$\delta = 0.1$		$\delta = 0.2$		$\delta = 0.3$		$\delta = 0.4$		$\delta = 0.5$	
Algorithm	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
MSE	0.0018	0.0003	0.0309	0.0094	0.0627	0.0217	0.1457	0.0523	0.2070	0.0586	0.3026	0.0859
LMLS	0.0029	0.0018	0.0148	0.0041	0.0177	0.0046	0.0310	0.0075	0.0429	0.0120	0.0630	0.0245
ILMedS	0.0032	0.0014	0.0056	0.0029	0.0069	0.0050	0.0299	0.0439	0.0270	0.0606	0.0401	0.0825

Table 2 The mean MSE for the networks trained to approximate function A (data with background noise)

	Data with background noise $\delta = 0.0$		$\delta = 0.1$		$\delta = 0.2$		$\delta = 0.3$		$\delta = 0.4$		$\delta = 0.5$	
Algorithm	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
MSE	0.0030	0.0054	0.0069	0.0056	0.0125	0.0034	0.0232	0.0062	0.0357	0.0098	0.0512	0.0099
LMLS	0.0037	0.0044	0.0055	0.0022	0.0087	0.0031	0.0166	0.0047	0.0276	0.0099	0.0387	0.0106
ILMedS	0.0035	0.0015	0.0043	0.0013	0.0055	0.0024	0.0055	0.0023	0.0065	0.0041	0.0092	0.0053

Table 3 The mean MSE for the networks trained to approximate function B (data with gross errors)

	Data with gross errors $\delta = 0.0$		$\delta = 0.1$		$\delta = 0.2$		$\delta = 0.3$		$\delta = 0.4$		$\delta = 0.5$	
Algorithm	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
MSE	0.0015	0.0004	0.0570	0.0331	0.1161	0.0729	0.1939	0.0848	0.2574	0.0817	0.3348	0.1777
LMLS	0.0015	0.0004	0.0073	0.0047	0.0167	0.0126	0.0419	0.0540	0.0780	0.0773	0.1139	0.0783
ILMedS	0.0037	0.0013	0.0052	0.0020	0.0091	0.0064	0.0141	0.0185	0.0259	0.0399	0.0245	0.0243

As one may notice, looking at the Tables 1 and 2, for the clean data without outliers, the traditional algorithm achieved the best results. It is not surprising because the MSE error function may be considered as optimal for the data without errors. The situation is different for the data containing artificially introduced outliers. In the case of data generated with the gross error model, the MSE-based method completely looses its efficiency. Even for 10 % of outliers, the mean error rises almost 16 times, and for a half of data substituted by outliers the error is over 160 times higher. Both robust algorithms perform significantly better: the mean error is the lowest for our novel ILMedS algorithm in each case. However, the LMLS algorithm has usually lower dispersion, measured by the standard deviation. The results of approximation for the maximum percentage of outliers are presented in the Fig. 2 (Table 9).

For the data with background noise the situation is similar. The new ILMedS algorithms outperforms other methods. The differences in performance between the worse and the best algorithm are not so significant in this case, while the difference between two robust method is larger. It means that the ILMedS algorithm is in this case much more robust to outliers than the LMLS method.

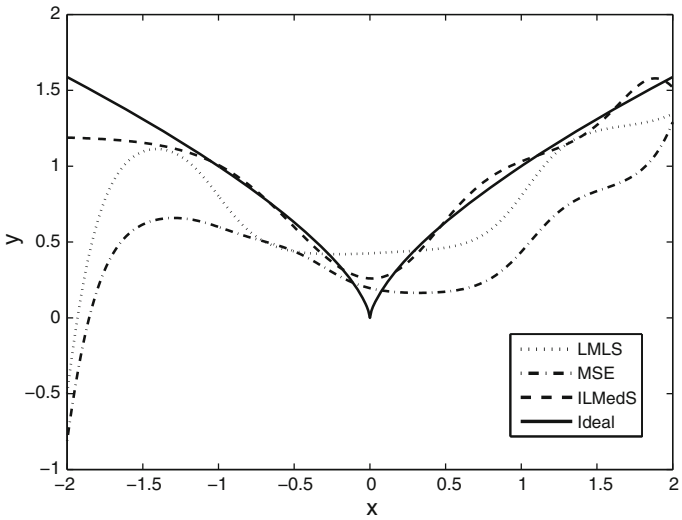

Fig. 2 Function approximation with different algorithms for the Gross Error Model, (function A), $\delta = 0.5$: backpropagation algorithm (*dashed-dotted line*), LMLS algorithm (*dotted line*), ILMedS algorithm (*dashed line*), ideal function (*solid line*)

Table 4 The mean MSE for the networks trained to approximate function B (data with background noise)

	Data with background noise $\delta = 0.0$		$\delta = 0.1$		$\delta = 0.2$		$\delta = 0.3$		$\delta = 0.4$		$\delta = 0.5$	
Algorithm	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
MSE	0.0016	0.0005	0.0077	0.0053	0.0172	0.0087	0.0315	0.0101	0.0432	0.0157	0.0700	0.0172
LMLS	0.0016	0.0005	0.0056	0.0041	0.0129	0.0065	0.0230	0.0077	0.0345	0.0137	0.0603	0.0175
ILMedS	0.0038	0.0019	0.0047	0.0014	0.0044	0.0019	0.0075	0.0038	0.0078	0.0052	0.0226	0.0221

Function B The second 1-D function to be approximated was a function considered in many articles [1,3] defined as:

$$y = \frac{\sin(x)}{x}. \tag{17}$$

The explanatory variable was sampled in the range $[-7.5, 7.5]$ with a step of 0.1. The network used to approximate this function had one input, ten hidden neurons with sigmoid transfer functions and one linear output neuron. The results gathered in the Tables 9 and 4 are similar to what could be observed for the previous approximation task: ILMedS algorithm outperforms two other methods for the network trained on the contaminated data set. When data without outliers were used, the traditional algorithm achieved the lowest averaged error.

6.3 Two-Dimensional Function Approximation

Function C The second approximation task was chosen following [6]. This 2-D function was previously used to test the SA-LMedS algorithm and various versions of the M-estimators based robust learning algorithms. The function to be approximated was given by:

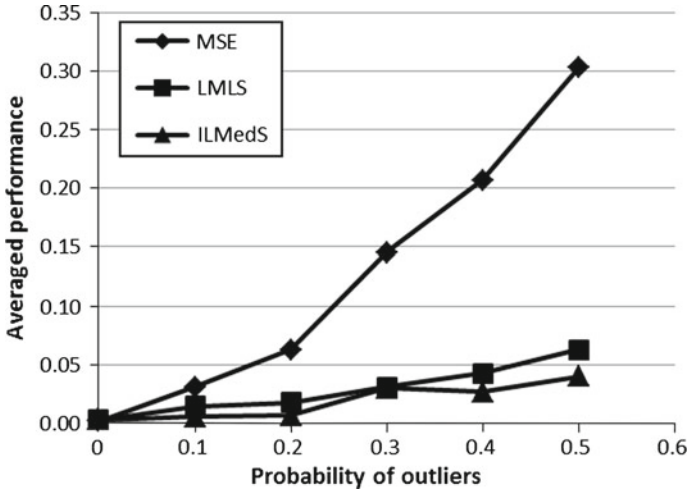

Fig. 3 Averaged performance of tested algorithms for the 1-D function approximation with Gross Error Model (function A)

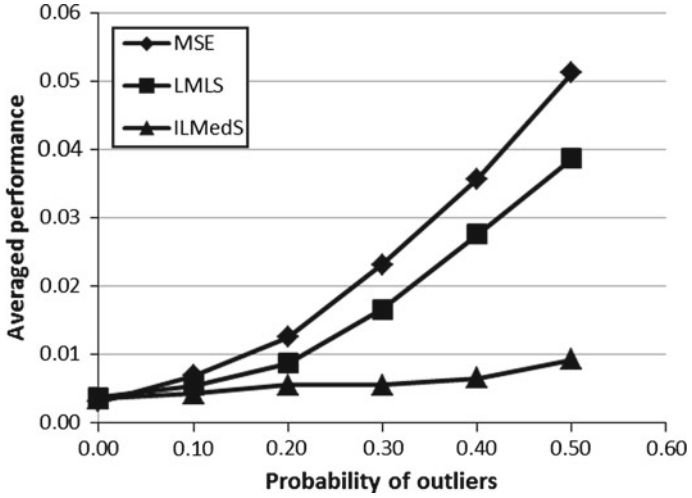

Fig. 4 Averaged performance of tested algorithms for the 1-D function approximation with background noise (function A)

$$y = x_1 e^{-\rho}, \tag{18}$$

where $\rho = x_1^2 + x_2^2$ and $x_1, x_2 \in [-2, 2]$. To create a data set, the function was sampled on the regular 16 x 16 grid. Then the errors according to one of the models were introduced, similarly to the 1-D case. The algorithms were employed to train a FFN with two inputs, ten hidden neurons with sigmoid transfer functions and one linear output neuron.

For the 2-D case the results are similar as in the 1-D: the best for the clean data appears the traditional MSE algorithm. Both robust methods have much better performance for the contaminated data sets as may be noticed in the Tables 5 and 6, and Figs. 6 and 7. Unlike to

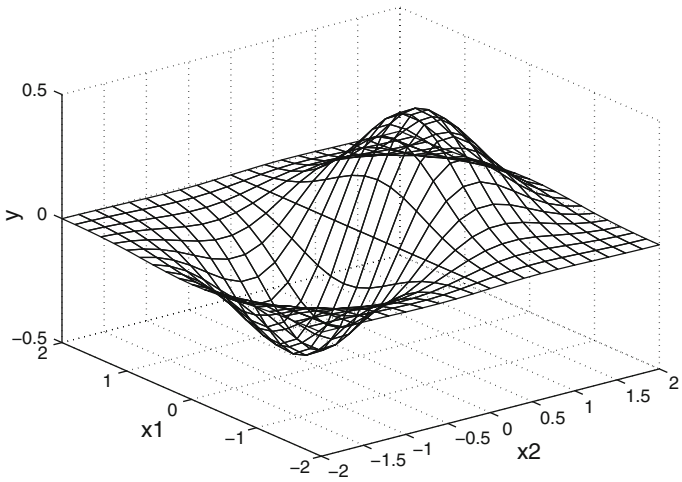

Fig. 5 2-D function to be approximated (function C)

Table 5 The mean MSE for the networks trained to approximate function C (data with gross errors)

	Data with gross errors $\delta = 0.0$		$\delta = 0.1$		$\delta = 0.2$		$\delta = 0.3$		$\delta = 0.4$		$\delta = 0.5$	
Algorithm	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
MSE	0.0018	0.0005	0.0127	0.0037	0.0224	0.0060	0.0398	0.0159	0.0537	0.0203	0.0775	0.0184
LMLS	0.0019	0.0007	0.0067	0.0017	0.0100	0.0024	0.0191	0.0092	0.0275	0.0105	0.0482	0.0201
ILMedS	0.0037	0.0013	0.0065	0.0025	0.0064	0.0026	0.0094	0.0036	0.0108	0.0036	0.0157	0.0097

Table 6 The mean MSE for the networks trained to approximate function C (data with background noise)

	Data with background noise $\delta = 0.0$		$\delta = 0.1$		$\delta = 0.2$		$\delta = 0.3$		$\delta = 0.4$		$\delta = 0.5$	
Algorithm	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
MSE	0.0018	0.0004	0.0167	0.0054	0.0275	0.0093	0.0404	0.0108	0.0541	0.0171	0.0746	0.0240
LMLS	0.0018	0.0005	0.0077	0.0027	0.0144	0.0048	0.0208	0.0094	0.0332	0.0146	0.0465	0.0167
ILMedS	0.0036	0.0012	0.0069	0.0031	0.0092	0.0068	0.0092	0.0040	0.0112	0.0046	0.0171	0.0108

the results for the 1-D function, the standard deviation for our robust ILMedS algorithm is smaller than for the LMLS method, while its performance is consistently better in each case.
Function D The last approximation task was a 2-D spiral defined as:

$$\begin{cases} x = \sin y \\ z = \cos y \end{cases} \tag{19}$$

Data points were generated by sampling the dependent variable in the range $[0, \pi]$ with a step $\pi/100$ and calculating x and z by the Eq. 19. The FFN was identical to the structure applied to approximate Function C. Looking at the Tables 7 and 8, one may notice that the

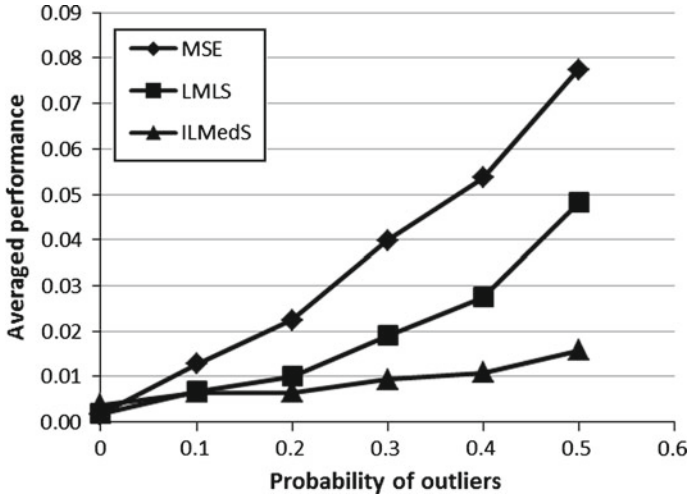

Fig. 6 Averaged performance of tested algorithms for the 2-D function approximation with Gross Error Model (function C)

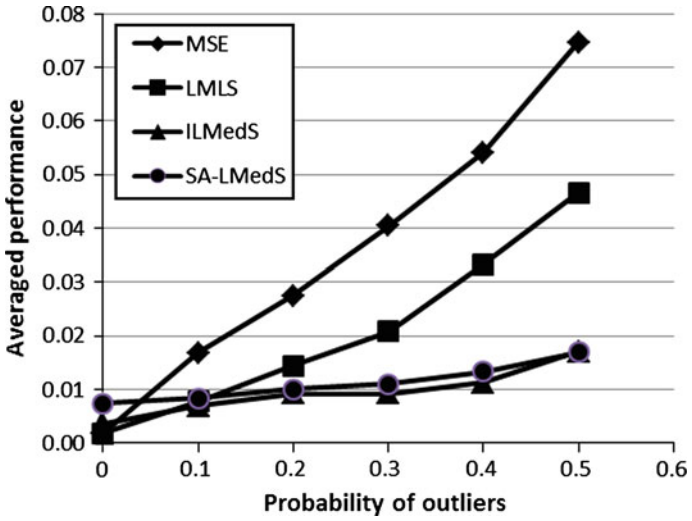

Fig. 7 Averaged performance of tested algorithms for the 2-D function approximation with background noise (function C)

average level of errors for the networks trained on contaminated data sets is significantly larger than in the former cases. For the clean training data, the performance of the ILMedS algorithm is worse than for the LMLS method but when the data are heavily corrupted by outliers, the novel algorithm acts again relatively better.

6.4 Time Performance

To compare the algorithms performance we used also another measure. The averaged times in seconds for the algorithms implemented in Matlab and ran on the Intel Core I5 CPU at

Table 7 The mean MSE for the networks trained to approximate function D (data with gross errors)

	Data with gross errors $\delta = 0.0$		$\delta = 0.1$		$\delta = 0.2$		$\delta = 0.3$		$\delta = 0.4$		$\delta = 0.5$	
Algorithm	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
MSE	0.0047	0.0019	0.2679	0.2244	0.5915	0.3580	0.8806	0.6075	1.1777	0.5818	1.5566	0.9333
LMLS	0.0050	0.0022	0.0382	0.0400	0.1202	0.2208	0.2290	0.2304	0.3437	0.3110	0.9674	0.5850
ILMedS	0.0169	0.0116	0.0517	0.0542	0.0465	0.0497	0.0694	0.0970	0.1560	0.2112	0.3366	0.6664

Table 8 The mean MSE for the networks trained to approximate function D (data with background noise)

	Data with background noise $\delta = 0.0$		$\delta = 0.1$		$\delta = 0.2$		$\delta = 0.3$		$\delta = 0.4$		$\delta = 0.5$	
Algorithm	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
MSE	0.0041	0.0018	0.0484	0.0448	0.1272	0.0661	0.2021	0.1134	0.2894	0.1405	0.3501	0.1016
LMLS	0.0041	0.0014	0.0238	0.0210	0.0824	0.0748	0.1121	0.0592	0.1803	0.1032	0.2936	0.1258
ILMedS	0.0133	0.0082	0.0189	0.0086	0.0273	0.0270	0.0668	0.0892	0.1109	0.1294	0.1832	0.1556

Table 9 Averaged time performance (in seconds) of the tested algorithms for 2-D function approximation

Algorithm	MSE	LMLS	ILMedS	SA-LMedS
Processing time	4.33	6.07	40.25	236.80

2.4GHz with 3GB RAM were gathered in the Table 9. These times can be considered as merely indicative, because the implementation was not optimized for speed. As it was expected, the fastest algorithm was the traditional one with the MSE error. The LMLS method needs 50 % more time to proceed, while the novel robust ILMedS algorithm runs ten times longer, being still better than the SA-LMedS algorithm with a processing time over 55 times higher than for the MSE method.

In the Fig. 7 additional plot for the SA-LMedS algorithm was shown. These averaged performances were obtained after setting its parameters to the values providing the lowest errors for several initial trails (only such method was proposed by the authors of the SA-LMedS algorithm [6]).

6.5 Summary of Results

Summarising the results of our experiments, one could formulate three general conclusions concerning tested algorithms:

1. The learning algorithm based on the MSE function is not reliable when the data are corrupted with outliers.
2. Both robust learning algorithms seem to perform better than the MSE method for contaminated data.
3. The ILMedS algorithm seems to outperform the LMLS algorithm for the data with gross errors.

These statements are obviously based on the averaged performance measured for our test sets. To convince the reader that they are more general, adequate statistical tests should be conducted. Following [5] we decided to use non-parametric tests, in this case namely, Friedman test and post-hoc Bonferroni–Dunn test.

Assuming significance level $\alpha = 0.05$, the Friedman test was performed on the experimental data to check whether the examined algorithms have different performances. Because the test detected a difference between the algorithms (the null-hypothesis about their identical performance was rejected), we could follow with the further analysis. We wanted to know whether the ILMedS algorithm performed better than two other methods. The post-hoc Bonferroni–Dunn test reveals only that at $\alpha = 0.05$ the ILMedS performance is significantly better than the performance of the MSE algorithm (such conclusion cannot be drawn for the LMLS algorithm). Moreover, in contrast to the ILMedS, the LMLS method is not significantly better than the MSE-based algorithm. At $\alpha = 0.1$ however, the difference between the ILMedS and LMLS is statistically important, so we can assume that the ILMedS outperforms also the LMLS method.

7 Conclusions

In this paper we presented novel robust learning algorithm based on the iterative Least Median of Squares. Our algorithm is not only robust to the presence of various amounts of outliers but also much faster and more precise than the SA-LMedS method known from the literature. The better performance of our method in comparison to other algorithms, in the presence of different types of outliers, was experimentally demonstrated. The model built by the network trained with the ILMedS learning algorithm is more accurate than for the traditional method, in the case of data contaminated with gross errors and outliers.

Future efforts may be directed to find faster and more effective way of minimizing the ILMedS error criterion function, e.g. by using some improved versions of the RBP method. Another possible solution could be a hybrid training method using both deterministic technique and stochastic algorithm.

References

1. Chen DS, Jain RC (1994) A robust back propagation learning algorithm for function approximation. IEEE Trans Neural Netw 5:467–479
2. Chuang C, Su S, Hsiao C (2000) The annealing robust backpropagation (ARBP) learning algorithm. IEEE Trans Neural Netw 11:1067–1076
3. Chuang CC, Jeng JT, Lin PT (2004) Annealing robust radial basis function networks for function approximation with outliers. Neurocomputing 56:123–139
4. David Sanchez VA (1995) Robustization of a learning method for RBFnetworks. Neurocomputing 9: 85–94
5. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:130
6. El-Melegy MT, Essai MH, Ali AA (2009) Robust training of artificial feedforward neural networks. Found Comput Intell 1:217–242
7. Hagan MT, Demuth HB, Beale MH (1996) Neural network design. PWS Publishing, Boston
8. Hampel FR, Ronchetti EM, Rousseeuw PJ, Stahel WA (1986) Robust statistics the approach based on influence functions. Wiley, New York

9. Haykin S (1999) Neural networks—a comprehensive foundation, 2nd edn. Prentice Hall, Upper Saddle River
10. Hornik K, Stinchconbe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2:359–366
11. Huber PJ (1981) Robust statistics. Wiley, New York
12. Igel C, Husken M (2003) Empirical evaluation of the improved Rprop learning algorithms. Neurocomputing 50:105–123
13. Jing X (2012) Robust adaptive learning of feedforward neural networks via LMI optimizations. Neural Netw 31:33–45
14. Liano K (1996) Robust error measure for supervised neural network learning with outliers. IEEE Trans Neural Netw 7:246–250
15. Olive DJ, Hawkins DM (2007) Robustifying robust estimators. Springer, New York
16. Pernia-Espinoza AV, Ordieres-Mere JB, Martinez-de-Pison FJ, Gonzalez-Marcos A (2005) TAO-robust backpropagation learning algorithm. Neural Netw 18:191–204
17. Riedmiller M, Braun H (1993) A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks. ICNN, San Francisco, pp 586–591
18. Rousseeuw PJ (1984) Least median of squares regression. J Am Stat Assoc 79:871–880
19. Rousseeuw AM, Leroy PJ (1987) Robust regression and outlier detection. Wiley, New York
20. Rusiecki AL (2005) Fault tolerant feedforward neural network with median neuron input function. Electron Lett 41(10):603–605
21. Rusiecki AL (2008) Robust MCD-based backpropagation learning algorithm. In: Rutkowski et al. (eds) ICAISC 2008. LNCS (LNAI), vol 5097. Springer, New York, pp 154–163
22. Sun S, Jin F (2011) Robust co-training. Int J Pattern Recognit Artif Intell 25(7):1113–1126