Abstract

In this paper, $M$-estimators, where $M$ stands for maximum likelihood, used in robust regression theory for linear parametric regression problems will be generalized to nonparametric maximum likelihood fuzzy neural networks (MFNNs) for nonlinear regression problems. Emphasis is put particularly on the robustness against outliers. This provides alternative learning machines when faced with general nonlinear learning problems. Simple weight updating rules based on gradient descent and iteratively reweighted least squares (IRLS) will be derived. Some numerical examples will be provided to compare the robustness against outliers for usual fuzzy neural networks (FNNs) and the proposed MFNNs. Simulation results show that the MFNNs proposed in this paper have good robustness against outliers.

Keywords: $M$-estimator; Fuzzy neural network (FNN); Maximum likelihood fuzzy neural networks (MFNN); Machine learning

1. Introduction

In a broad range of practical applications, data collected inevitably contain one or more atypical observations called outliers; that is, observations that are well separated from the majority or bulk of the data, or in some fashion deviate from the general pattern of the data. As is well known in linear regression theory, classical least squares fit of a regression model can be very adversely influenced by outliers, even by a single one, and often fails to provide a good fit to the bulk of the data [17]. Robust regression that is resistant to the adverse effects of outlying response values offers a half-way house between including outliers and omitting them entirely. Rather than omitting outliers, it dampens their influence on the fitted regression curve by down-weighting them. It is desirable that the robust estimates provide a good fit for the majority of the data when the data contain outliers, as well as when the data are free of them. One of the main approaches to robust regression involves $M$-estimation [5,6,9,10,17,21]. A regressor or a learning machine is said to be robust if it is not sensitive to outliers in the data.

Machine learning, namely learning from examples, has been an active research area for several decades. Fuzzy neural network (FNN) has long been reckoned as a successful learning machine for many problems in science and engineering. The class of approximating functions represented as fuzzy neural networks, with the number of hidden
nodes unfixed, possesses the universal approximation property, i.e., they are universal approximators [12,24]. The universal approximation property is crucial for the success of a learning machine in a variety of applications. Moreover, FNNs are nonparametric in the sense that they do not make any assumptions of the functional form, e.g., linearity, of the final predictive functions. This provides a great deal of flexibility in designing an appropriate learning machine for the problem at hand.

Based on the theory of robust statistics, the Hampel’s hyperbolic tangent cost function was adopted for function approximation using artificial neural networks in [1,8]. The back propagation (BP) learning algorithms were proposed in their papers. The concept of annealing was used in the BP learning algorithm to improve the convergence efficiency in [4]. Other architectures such as radial basis function (RBF) networks with robust criteria were developed, in which sequences of sigmoidal functions and robust object function were adopted to overcome the problem of outliers [14]. Later, the annealing method was also applied to RBF networks to improve the performance for function approximation with outliers [2].

A fuzzy neural network (FNN) is a fuzzy system represented as a neural network. In order to overcome the problems of function approximation for a nonlinear system with noise and outliers, a robust neural-fuzzy clustering method was proposed in [19]. Wang et al. [23] proposed BP learning algorithms for FNNs with B-spline membership functions and the Hampel’s cost function. In their paper, the authors used the learning mechanism to tune both the parameters of B-spline membership functions and the weights of defuzzification. To reduce the training complexity of Wang’s algorithm, Tsai et al. [20] proposed an optimal design of FNN by determining the optimal learning rates. Note that the learning rules proposed in the aforementioned papers are essentially conventional learning algorithms based on the gradient decent method, in which high complexity is the main drawback. Related studies on robust TSK fuzzy modeling were proposed in [3,15,25]. Recently, the Wilcoxon fuzzy neural networks with rank-based cost function were proposed aiming also at improving the robustness against outliers [7].

The main aim of this study is to generalize M-estimators used in robust regression theory for linear parametric regression problems to nonparametric MFNNs for nonlinear regression problems. Emphasis is put particularly on the robustness against outliers. Though we consider only MFNNs in this study, the extension of our approach to other neural networks such as artificial neural networks and radial basis function networks is straightforward.

2. M-estimator

In this section, we briefly review the M-estimation for linear parametric regression problems. For more background materials, see, e.g. [6,10,17] and the references therein.

The general multiple regression model with $k$ predictor variables is given by

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_j x_{ij} + \cdots + \beta_k x_{ik} + \varepsilon_i, \quad i \in \mathbb{n}. \quad (1)$$

Here, $y_i$ is the response value for the $i$th case, $x_{ij}, i \in \mathbb{n}, j \in \mathbb{k}$, is the value of the $j$th predictor variable for the $i$th case, $\varepsilon_i$ is the random error term, and $\beta_0, \beta_1, \ldots, \beta_k$ are unknown parameters to be estimated. The model (1) can be written as

$$y_i = x_i^T \beta + \varepsilon_i, \quad i \in \mathbb{n},$$

where

$$\beta = [\beta_0 \ \beta_1 \ \cdots \ \beta_k]^T, \quad x_i^T = [1 \ x_{i1} \ \ldots \ x_{ik}].$$
Let $b$ be an estimate of $\beta$. Then the $i$th fitted value $\hat{y}_i$ and residual $e_i$, $i \in \mathbb{N}$, are defined by, respectively,

$$\hat{y}_i = x_i^T b, \quad e_i = y_i - \hat{y}_i.$$  

The $M$-estimate $\hat{\beta}$ is defined to be a minimizer of the following robust criterion function:

$$\sum_{i=1}^{n} \rho \left( \frac{e_i(\beta)}{s} \right),$$  \hspace{1cm} (2)

where $\rho(\cdot)$ is usually called a $\rho$-function and $s$ is a robust estimate of scale. A popular choice for $s$ is the normalized median absolute deviation about the median $[13,18]$

$$s = \frac{\text{median}(|e_i - \text{median}(e_i)|)}{0.6745}. \hspace{1cm} (3)$$

Some commonly used $\rho$-functions in (2) include Tukey’s biweight (or bisquare) function, Huber’s $t$ function, Hampel’s $17A$ function, Ramsay’s $E_a$ function, and Andrews’ wave function $[17,18]$. The approach used in the following development applies to any $\rho$-functions.

In the illustrative examples of this study, we will use the following two $\rho$-functions:

- Huber’s $t$ function ($t$: a positive constant)
  \[
  \rho(u) := \begin{cases} 
  u^2/2, & |u| \leq t, \\
  t|u| - t^2/2, & |u| > t, 
  \end{cases} \quad u \in \mathbb{R}.
  \]

- Ramsay’s $E_a$ function ($a$: a positive constant)
  \[
  \rho(u) := \frac{1}{a^2} [1 - \exp(-a|u|) \cdot (1 + a|u|)], \quad u \in \mathbb{R}.
  \]

### 3. FNN

Consider the standard fuzzy system as shown in Fig. 1, where $U$ is the input space and $V$ is the output space. It consists of four principal components, namely fuzzy rule base, fuzzy inference engine, fuzzifier, and defuzzifier. The fuzzy rule base consists of a family of IF–THEN rules reflecting the human knowledge of the system. The main task of the fuzzy inference engine is to combine those IF–THEN rules in order to map a fuzzy set in $U$ to a fuzzy set in $V$. To provide the interface with the crisp environment, the fuzzifier transforms a crisp point in $U$ to a fuzzy set in $U$, whilst the defuzzifier specifies a fuzzy set in $V$ to a crisp point in $V$. Hence, the fuzzy system is in fact a crisp nonlinear map from the input space $U$ to the output space $V$ $[22]$.

Consider a fuzzy system with $n$ inputs and $p$ outputs. The fuzzy rule base is composed of $m$ fuzzy rules in canonical form:

$$R_j: \text{IF } x_1 \text{ is } A_{1j} \text{ and } x_2 \text{ is } A_{2j} \text{ and } \ldots \text{ and } x_n \text{ is } A_{nj}, \quad \text{THEN } y_1 \text{ is } B_{j1} \text{ and } y_2 \text{ is } B_{j2} \text{ and } \ldots \text{ and } y_p \text{ is } B_{jp},$$

![Fig. 1. Standard fuzzy system.](image-url)
where \( j \in m \). There are many possible choices for the membership functions of fuzzy sets \( A_{ij} \) and \( B_{jk} \), fuzzifier, inference engine, and defuzzifier, resulting in different fuzzy systems. In the following development, we will, for simplicity, consider only one commonly used fuzzy system.

Consider, in particular, the fuzzy system with singleton fuzzifier, product inference engine, center-average defuzzifier, and normal Gaussian membership functions. Then, the fuzzy system \( f \) is a nonlinear map given by

\[
y_k = f_k(x) = \frac{\sum_{j=1}^{m} w_{jk} \exp[-\sum_{i=1}^{n} (x_i - c_{ij})^2/v_{ij}]}{\sum_{j=1}^{m} \exp[-\sum_{i=1}^{n} (x_i - c_{ij})^2/v_{ij}]}, \quad k \in p,
\]

\( x = [x_1 \ldots x_n]^T \in \mathbb{R}^n \),

where \( w_{jk} \) is the center of the normal fuzzy set \( B_{jk} \), and \( c_{ij} \) and \( v_{ij} \) are the center and “variance”, respectively, of the Gaussian fuzzy set \( A_{ij} \) [22].

Define, for \( i \in n, j \in m, \) and \( k \in p, \)

\[
u_j = \sum_{i=1}^{n} (x_i - c_{ij})^2/v_{ij}, \quad r_j = \exp(-u_j),
\]

\[
s_k = \sum_{j=1}^{m} w_{jk} r_j, \quad g = \sum_{j=1}^{m} r_j,
\]

then

\[
y_k = s_k/g.
\]

The fuzzy system (4) or (5) can be represented as a feedforward network, namely, the fuzzy neural network (FNN), as shown in Fig. 2 [11,22]. The advantage of using FNN for machine learning is that the parameters \( w_{jk}, c_{ij}, \) and \( v_{ij} \) usually have clear physical meanings and we have some intuitive methods to choose good initial values for them.

![Fig. 2. Fuzzy neural network.](image)
Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^p$. Suppose we are given the training data set

$$S := \{(x_q, d_q)\}_{q=1}^l \subseteq X \times Y.$$  \hfill (6)

In the following, we will use the subscript $q$ to denote the $q$th example. For instance, $x_{qi}$ denotes the $i$th component of the $q$th pattern $x_q \in \mathbb{R}^n$, $q \in l$, $i \in n$.

The residual $e_{qk}$ at the $k$th output node due to the $q$th example is defined by

$$e_{qk} = d_{qk} - y_{qk}, \quad q \in l, \ k \in p.$$  \hfill (7)

In a standard FNN, the goal is to choose weights $w_{jk}$, $c_{ij}$, and $v_{ij}$ that minimize the total sum of squared errors

$$E_{\text{total}} := \frac{1}{2} \sum_{k=1}^p \sum_{q=1}^l e_{qk}^2.$$  \hfill (8)

Appropriate updating rules for minimizing (8), e.g., back propagation (BP) algorithms, can be used to approximate the optimal weights.

4. MFNN

Given the training data set (6), the goal of MFNN is to choose weights that minimize

$$\Psi_{\text{total}} := \sum_{k=1}^p \Psi_k, \quad \Psi_k := \sum_{q=1}^l \rho \left( \frac{e_{qk}}{c_k} \right), \quad k \in p.$$  \hfill (9)

where $c_k > 0$, $k \in p$, is a robust estimate of scale. As in (3), $c_k$ may be computed by

$$c_k = \text{median}_q \{|e_{qk} - \text{median}_q[e_{qk}]|/0.6745 \}.$$  \hfill (10)

From (7) and (9), we have

$$\Psi_{\text{total}} = \sum_{k=1}^p \Psi_k, \quad \Psi_k := \sum_{q=1}^l \rho \left( \frac{e_{qk}}{c_k} \right), \quad k \in p.$$  \hfill (11)

Now, we introduce an incremental gradient descent algorithm. In this algorithm, $\Psi_k$‘s in (11) are minimized in sequence. The updating rules are given by

$$w_{jk} \leftarrow w_{jk} - \eta_w \frac{\partial \Psi_k}{\partial w_{jk}}$$

$$= w_{jk} + \eta_w \frac{1}{c_k} \sum_{q=1}^l \rho' \left( \frac{e_{qk}}{c_k} \right) \frac{r_{qj}}{g_q},$$  \hfill (12a)

$$c_{ij} \leftarrow c_{ij} - \eta_c \frac{\partial \Psi_k}{\partial c_{ij}}$$

$$= c_{ij} + \eta_c \frac{1}{c_k} \sum_{q=1}^l \rho' \left( \frac{e_{qk}}{c_k} \right) \left( w_{jk} - \frac{s_{qj}}{g_q} \right) \frac{r_{qj}}{g_q} \cdot 2 \frac{(x_{qi} - c_{ij})}{v_{ij}},$$  \hfill (12b)

$$v_{ij} \leftarrow v_{ij} - \eta_v \frac{\partial \Psi_k}{\partial v_{ij}}$$

$$= v_{ij} + \eta_v \frac{1}{c_k} \sum_{q=1}^l \rho' \left( \frac{e_{qk}}{c_k} \right) \left( w_{jk} - \frac{s_{qj}}{g_q} \right) \frac{r_{qj}}{g_q} \cdot \left( \frac{x_{qi} - c_{ij}}{v_{ij}} \right)^2,$$  \hfill (12c)

where $\eta_w, \eta_c, \eta_v > 0$ are learning rates.
In (Gaussian) regression problem, we may have a better way to update $w_{jk}$ based on iteratively reweighted least squares (IRLS). However, $c_{ij}$ and $v_{ij}$ will still be updated according to (12b) and (12c), respectively.

Define

$$w_k := [w_{1k} \ldots w_{mk}]^T \in \mathbb{R}^m, \quad r_q := [r_{q1} \ldots r_{qm}]^T \in \mathbb{R}^m.$$ (13)

Then we have

$$y_{qk} = w_k^T r_q / g_q.$$ (14)

By setting $\partial \Psi_k / \partial w_{jk} = 0$, the estimating equations are given by

$$0 = \sum_{q=1}^l \rho'(d_{qk} - (w_k^T r_q / g_q)) / c_k \left[ 1 - c_k (d_{qk} - (w_k^T r_q / g_q)) r_{qj} / g_q \right] r_{qj} / g_q, \quad j \in m,$$

or

$$\sum_{q=1}^l \tilde{z}_{qk} (d_{qk} - (w_k^T r_q / g_q)) r_{qj} / g_q = 0, \quad j \in m,$$

where, for $k \in p, j \in m$,

$$\tilde{z}_{qk} = \begin{cases} 1, & d_{qk} = w_k^T r_q / g_q, \\ \rho'(d_{qk} - (w_k^T r_q / g_q)) / c_k, & d_{qk} \neq w_k^T r_q / g_q. \end{cases}$$ (15)

Thus, we may convert the problem of minimizing $\Psi_k$ with respect to $w_{jk}$ to a weighted least squares (WLS) problem. That is, we seek $w_{jk}$ to minimize

$$E_k(w_k) := 2^{-1} \sum_{q=1}^l \tilde{z}_{qk} e_{qk}^2 = 2^{-1} \sum_{q=1}^l \tilde{z}_{qk} (d_{qk} - (w_k^T r_q / g_q))^2,$$ (16)

where

$$e_{qk} := d_{qk} - (w_k^T r_q / g_q), \quad q \in \mathbb{L}.$$ (17)

This can be verified as follows. By setting $\partial E_k / \partial w_{jk} = 0$, we have

$$0 = \sum_{q=1}^l \tilde{z}_{qk} (d_{qk} - (w_k^T r_q / g_q)) r_{qj} / g_q, \quad j \in m,$$

which are the same as the estimating equations obtained earlier. Any efficient recursive weighted least squares algorithm can be used to solve the optimization problem (16).

Define

$$\theta_q := r_q / g_q := [r_{q1} / g_q \ldots r_{qm} / g_q]^T \in \mathbb{R}^m, \quad q \in \mathbb{L}.$$ (18)

Based on the analysis above, we now propose an IRLS algorithm for updating $w_{jk}$, which can be stated as follows:

Data: Training set $\tilde{S} := \{ (\theta_q, d_{qk}) \}_{q=1}^l$

Goal: Find $w_{jk}, \quad j \in m$, that minimize $\Psi_k$ in (11).

Step 1: Find initial robust estimates $w_{jk}, \quad j \in m$.

Step 2: Calculate the predicted outputs $y_{qk}, \quad q \in \mathbb{L}$, defined in (13) and (14).
Step 3: Calculate the residuals $e_{qk}$, $q \in I$, in (17).
Step 4: Compute $c_k$ in (10).
Step 5: Find the weights $\hat{e}_{qk}$, $q \in I$, in (15).
Step 6: The new $w_{jk}$, $j \in m_i$, are obtained by minimizing $E_k(w_k)$ in (16).
Step 7: If the stopping criterion is met, then stop; otherwise, go to Step 2.

Note that in Step 1 we may use LAD (least absolute deviations) or Wilcoxon estimates as initial estimates of $w_{jk}$, $j \in m_i$. Note also that the dispersion estimate $c_k$ is updated at each iteration in the preceding algorithm.

5. Illustrative examples

In this section, we compare the performances of the standard FNNs and the proposed MFNNs for several illustrative nonlinear regression problems. Emphasis is put particularly on the robustness against outliers.

In the following simulations of Examples 1 and 2, the uncorrupted training data set consists of 50 randomly chosen x-points (training patterns) with the corresponding y-values (target values) evaluated from the underlying true functions.
The corrupted training data set is composed of the same $x$-points as the corresponding uncorrupted ones but with 40% randomly chosen $y$-values corrupted by adding random values from a uniform distribution defined on $[-1, 1]$.

For simple comparison purpose, we first set (after some trials) the numbers of hidden nodes for both FNN and MFNN so that the uncorrupted training data will not be overfitted. Then we use the corrupted training data to test the robustness against outliers of these learning machines.

For the sake of notation simplicity, in the following development we will use HFNN and RFNN to denote the MFNNs with Huber’s $t$ function and Ramsay’s $E_a$ function, respectively.

In the following simulations of Examples 1–4, the initial values of $c_{ij}$ and $v_{ij}$ are set to be the sample mean and sample variance, respectively, of observed values of the predictor $x_i$ for all $j \in m$, and the initial values of $w_{jk}$ are all set to be zero.

**Example 1.** Suppose the true regression function is given by the sinc function

$$y = \begin{cases} 
1, & x = 0, \\
\sin(x)/x, & x \neq 0,
\end{cases} \quad x \in [-10, 10].$$
Table 1
Simulations for FNNs and MFNNs in Example 3 (geophones data).

<table>
<thead>
<tr>
<th>No. of hidden nodes</th>
<th>FNN</th>
<th></th>
<th>HFNN ((t=3.0))</th>
<th></th>
<th>RFNN ((a=0.1))</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>27.126</td>
<td>3.388</td>
<td>6.846</td>
<td>2.094</td>
<td>6.826</td>
<td>2.090</td>
</tr>
<tr>
<td>5</td>
<td>12.207</td>
<td>2.356</td>
<td>6.558</td>
<td>2.050</td>
<td>6.574</td>
<td>2.052</td>
</tr>
<tr>
<td>7</td>
<td>9.900</td>
<td>2.146</td>
<td>5.787</td>
<td>1.898</td>
<td>5.796</td>
<td>1.906</td>
</tr>
<tr>
<td>8</td>
<td>13.245</td>
<td>2.243</td>
<td>5.757</td>
<td>1.891</td>
<td>5.761</td>
<td>1.894</td>
</tr>
<tr>
<td>9</td>
<td>17.718</td>
<td>2.710</td>
<td>5.754</td>
<td>1.898</td>
<td>5.762</td>
<td>1.901</td>
</tr>
<tr>
<td>10</td>
<td>29.046</td>
<td>3.390</td>
<td>5.808</td>
<td>1.894</td>
<td>5.798</td>
<td>1.895</td>
</tr>
</tbody>
</table>

For uncorrupted data shown in Fig. 3(a), FNN (with nine hidden nodes) and HFNN (with nine hidden nodes and \(t=0.4\)) estimates are almost indistinguishable from the true function, and the uncorrupted training data are not overfitted. For the 40% corrupted data shown in Fig. 3(b), the FNN estimate is apparently affected by outliers resulting in an oscillatory fitted curve, whereas HFNN estimate is much less affected by outliers and thus is robust. The performance of RFNN is similar to that of HFNN.

Example 2. Suppose the true regression function is given by the Hermite function

\[
y = 1.1 \cdot (1 - x + 2x^2) \cdot e^{-x^2/2}, \quad x \in [-5,5].
\]

For uncorrupted data shown in Fig. 4(a), FNN (with nine hidden nodes) and RFNN (with nine hidden nodes and \(a=3.0\)) estimates are almost indistinguishable from the true function, and the uncorrupted training data are not overfitted. For the 40% corrupted data shown in Fig. 4(b), the FNN estimate is apparently affected by outliers. On the other hand, the RFNN estimate is less affected by those outliers and outperforms the FNN estimate. The performance of HFNN is similar to that of RFNN.

The advantage of using simulated data is that we can see how close our models come to the truth. Certainly, one could argue that the efficacy of MFNNs in Examples 1 and 2 was tested only on contrived problems with known true regression functions. This means that the structure of the underlying process and the correct values of the output were
known. Of course, we usually do not have such fortune in most real-world problems. In the following, we will provide a real-world problem to illustrate the use of MFNN. For better visualization, we choose the problem with low input dimension in the next example.

**Example 3 (Maindonald and Braun [16])**. The “geophones” data frame in R-package DAAG has 56 rows and 2 columns. The geophones data are listed as follows:

<table>
<thead>
<tr>
<th>Distance</th>
<th>Thickness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.75</td>
</tr>
<tr>
<td>2</td>
<td>15.00</td>
</tr>
<tr>
<td>3</td>
<td>16.25</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>54</td>
<td>81.25</td>
</tr>
<tr>
<td>55</td>
<td>82.50</td>
</tr>
<tr>
<td>56</td>
<td>83.75</td>
</tr>
</tbody>
</table>

Fig. 6. Fitted curves of Example 3: (a) FNN and HFNN, (b) FNN and RFNN.
Table 2
Description of data sets in Example 4.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of cases</th>
<th>Number of predictors</th>
<th>R-package</th>
</tr>
</thead>
<tbody>
<tr>
<td>cars</td>
<td>50</td>
<td>1</td>
<td>datasets</td>
</tr>
<tr>
<td>dewpoint</td>
<td>72</td>
<td>2</td>
<td>DAAG</td>
</tr>
<tr>
<td>ustemp</td>
<td>56</td>
<td>2</td>
<td>SemiPar</td>
</tr>
</tbody>
</table>

Table 3
Simulations for FNNs and MFNNs in Example 4 (cars data).

<table>
<thead>
<tr>
<th>No. of hidden nodes</th>
<th>FNN Testing MSE</th>
<th>HFNN (t=1.0) Testing MSE</th>
<th>RFNN (a=0.3) Testing MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Testing MAD</td>
<td>Testing MAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Testing MAD</td>
<td>Testing MAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Testing MAD</td>
<td>Testing MAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Testing MAD</td>
<td>Testing MAD</td>
</tr>
<tr>
<td>4</td>
<td>400.407</td>
<td>15.750</td>
<td>341.082</td>
</tr>
<tr>
<td>5</td>
<td>427.600</td>
<td>16.458</td>
<td>335.702</td>
</tr>
<tr>
<td>6</td>
<td>467.723</td>
<td>16.841</td>
<td>309.955</td>
</tr>
<tr>
<td>7</td>
<td>468.802</td>
<td>16.640</td>
<td>312.128</td>
</tr>
<tr>
<td>8</td>
<td>515.125</td>
<td>17.903</td>
<td>310.828</td>
</tr>
<tr>
<td>9</td>
<td>498.048</td>
<td>17.784</td>
<td>307.485</td>
</tr>
<tr>
<td>10</td>
<td>469.186</td>
<td>17.720</td>
<td>314.593</td>
</tr>
</tbody>
</table>

Table 4
Simulations for FNNs and MFNNs in Example 4 (dewpoint data).

<table>
<thead>
<tr>
<th>No. of hidden nodes</th>
<th>FNN Testing MSE</th>
<th>HFNN (t=1.0) Testing MSE</th>
<th>RFNN (a=0.3) Testing MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Testing MAD</td>
<td>Testing MAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Testing MAD</td>
<td>Testing MAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Testing MAD</td>
<td>Testing MAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Testing MAD</td>
<td>Testing MAD</td>
</tr>
<tr>
<td>4</td>
<td>7.703</td>
<td>1.399</td>
<td>0.250</td>
</tr>
<tr>
<td>5</td>
<td>4.368</td>
<td>0.969</td>
<td>0.230</td>
</tr>
<tr>
<td>6</td>
<td>4.531</td>
<td>0.962</td>
<td>0.198</td>
</tr>
<tr>
<td>7</td>
<td>4.213</td>
<td>0.962</td>
<td>0.228</td>
</tr>
<tr>
<td>8</td>
<td>3.826</td>
<td>0.929</td>
<td>0.210</td>
</tr>
<tr>
<td>9</td>
<td>5.054</td>
<td>1.057</td>
<td>0.223</td>
</tr>
<tr>
<td>10</td>
<td>8.900</td>
<td>1.546</td>
<td>0.244</td>
</tr>
</tbody>
</table>

In this seismic timing data, the response variable “thickness” denotes the time for signal to pass through substratum and the predictor variable “distance” represents the location of geophone.

To evaluate the performance on generalization capability of FNNs and MFNNs, we will repeat the following training and testing procedure for 10 runs. In each run, we randomly select 37 data (about 2/3 of the total data) to form the training set with the remaining 19 data as the testing data. In each run, the testing error rates are expressed in terms of the mean squared error (MSE) and the mean absolute deviation (MAD). We then average those testing error rates to give estimated testing error rates. Note that the MFNN is not aiming at minimizing either MSE or MAD in the training process.

The simulation results are shown in Table 1. To give better vision, Fig. 5 shows, for instance, the testing MAD versus the number of hidden nodes for geophones data. From Table 1 and Fig. 5, it is seen that MFNNs usually outperform FNNs. The fitted curves of FNN, HFNN, and RFNN, all with six hidden nodes, using full training data are shown in Fig. 6. As can be observed, HFNN and RFNN estimates provide better fits to the data.

Example 4. To further illustrate the use of MFNNs to more real-world problems and to compare the performances with the corresponding FNNs, we consider in this example three real-world data sets. These data sets are described in Table 2, in which the sources are R-packages “datasets”, “DAAG”, and “SemiPar”. Using the same testing procedure
Table 5
Simulations for FNNs and MFNNs in Example 4 (ustemp data).

<table>
<thead>
<tr>
<th>No. of hidden nodes</th>
<th>FNN</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>4</td>
<td>49.317</td>
<td>4.686</td>
<td>15.079</td>
<td>2.950</td>
<td>13.891</td>
<td>2.806</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>30.037</td>
<td>3.693</td>
<td>10.256</td>
<td>2.365</td>
<td>11.420</td>
<td>2.469</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>36.942</td>
<td>4.056</td>
<td>9.287</td>
<td>2.344</td>
<td>8.891</td>
<td>2.313</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>41.039</td>
<td>4.325</td>
<td>7.793</td>
<td>2.221</td>
<td>9.067</td>
<td>2.353</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>50.161</td>
<td>4.622</td>
<td>8.970</td>
<td>2.336</td>
<td>9.048</td>
<td>2.337</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>73.664</td>
<td>6.022</td>
<td>9.696</td>
<td>2.394</td>
<td>12.741</td>
<td>2.506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>86.602</td>
<td>6.178</td>
<td>12.570</td>
<td>2.546</td>
<td>13.755</td>
<td>2.739</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

in Example 3, the simulation results for the three data sets are summarized in Tables 3–5. From the tables, it is observed that MFNNs do outperform FNNs both in terms of testing MSE and MAD.

6. Discussion and conclusion

The M-estimators used in robust regression theory for linear parametric regression problems was generalized in this study to nonparametric maximum likelihood fuzzy neural networks for nonlinear regression problems. This provides alternatives to currently popular learning machines. Simple weight updating rules based on gradient descent and iteratively reweighted least squares were derived. Some numerical examples were provided to compare the robustness against outliers for standard FNNs and MFNNs. Simulation results showed that the MFNNs proposed in this paper have good robustness against outliers.

As is well known in statistical regression theory, two important properties of robust estimators are breakdown and efficiency. From a practical viewpoint, it is important to know the finite-sample breakdown point and finite-sample efficiency of a particular robust estimator. Roughly speaking, the finite-sample breakdown point of an estimator is the smallest fraction of anomalous data that can cause the estimator to be useless, and finite-sample efficiency measures how well an estimator works with reference to the ordinary least squares on “clean” data for sample sizes consistent with those of interest in the problem at hand [18]. It is an interesting research topic to know the finite-sample breakdown point and finite-sample efficiency of the MFNNs proposed in this paper and to compare them with the corresponding linear M-estimators.

Acknowledgements

The authors would like to thank the anonymous referees for their valuable comments, which helped the authors improve the clarity of this paper. The research reported here was supported by the National Science Council, Taiwan, under Grant no. NSC97-2221-E-214-070-MY2.

References