Improving the Classification Performance of Optimal Linear Associative Memory in the Presence of Outliers

Ana Luiza Bessa de Paula Barros[1,2] and Guilherme A. Barreto[2]

[1] Department of Computer Science, State University of Ceará
Campus of Itaperi, Fortaleza, Ceará, Brazil
analuiza@larces.uece.br
[2] Department of Teleinformatics Engineering, Federal University of Ceará
Center of Technology, Campus of Pici, Fortaleza, Ceará, Brazil
guilherme@deti.ufc.br

Abstract. The optimal linear associative memory (OLAM) proposed by Kohonen and Ruohonen [16] is a classic neural network model widely used as a standalone pattern classifier or as a fundamental component of multilayer nonlinear classification approaches, such as the extreme learning machine (ELM) [10] and the echo-state network (ESN) [6]. In this paper, we develop an extension of OLAM which is robust to labeling errors (outliers) in the data set. The proposed model is robust to label noise not only near the class boundaries, but also far from the class boundaries which can result from mistakes in labelling or gross errors in measuring the input features. To deal with this problem, we propose the use of M-estimators, a parameter estimation framework widely used in robust regression, to compute the weight matrix operator, instead of using the ordinary least squares solution. We show the usefulness of the proposed classification approach through simulation results using synthetic and real-world data.

Keywords: Linear Associative Memory, Moore-Penrose Generalized Inverse, Pattern Classification, Outliers, M-Estimation.

1 Introduction

The OLAM model, as proposed by Kohonen and Ruohonen [16], is a well known computational paradigm of associative memory. As such, information in OLAM is stored distributively in a matrix operator, so that it can recall a stored data by specifying all or portion of a key (degraded key). The OLAM has the property of providing rapid recall of information, and it can tolerate local damage without a great degradation in performance.

Previous studies have evaluated empirically and/or theoretically the robustness of OLAM to noisy input patterns [3,19,20]. The main conclusion that these works have pointed out is that when input key vectors are degraded (noisy), the

I. Rojas, G. Joya, and J. Cabestany (Eds.): IWANN 2013, Part I, LNCS 7902, pp. 622–632, 2013.

model becomes extremely sensitive (unstable) and its association error becomes unacceptably large. Authors have tackled this limitation of the OLAM by including nonlinear features into the associative memory model [13] or by taking into consideration the properties of the noise directly into the developing of the model [1].

In this paper, we are interested in OLAM not for associative memory, but rather for pattern classification. In this context, the OLAM is theoretically equivalent to the least-squares classifier [4,21] and has been used either as a standalone classifier [2,5,15] or as a fundamental building block of multilayer nonlinear classification approaches, such as the radial basis functions network [18], the extreme learning machine (ELM) [10] and the echo-state network (ESN) [6].

In many real-world classification problems the labels provided for the data are noisy. There are typically two kinds of noise in labels. Noise near the class boundaries often occurs because it is hard to consistently label ambiguous data points. Labelling errors far from the class boundaries can occur because of mistakes in labelling or gross errors in measuring the input features. Labelling errors far from the boundary comprises a particular category of *outliers* [14].

In order to allow the OLAM classifier to handle outliers efficiently, in this paper we propose the use of M-estimators [12], a broad framework widely used for parameter estimation in robust regression problems, to compute the weight matrix operator instead of using the ordinary least squares solution. We show through simulations on synthetic and real-world data that the resulting OLAM classifier is very robust to outliers.

Despite the fact that M-estimation has been widely used in regression problems (see, e.g. references [9,17]), its application to supervised pattern classification problems is much less studied. In reality, we were not able to find a single paper on the combined use of M-estimation and neural network classifiers. Furthermore, to the best of our knowledge, this is the first time the performance of the OLAM model as a classifier is evaluated under the presence of outliers.

The remainder of the paper is organized as follows. In Section 2, we briefly review the fundamentals of OLAM in the context of pattern classification. Then, in Section 3 we describe the basic ideas and concepts behind the M-estimation framework and introduce our approach to robust supervised pattern classification using OLAM. In Section 4 we present the computer experiments we carried out using synthetic and real-world datasets and also discuss the achieved results. The paper is concluded in Section 5.

2 Fundamentals of OLAM

Let us assume that N data pairs $\{(\mathbf{x}_\mu, \mathbf{d}_\mu)\}_{\mu=1}^{N}$ are available for building and evaluating the model, where $\mathbf{x}_\mu \in \mathbb{R}^{p+1}$ is the μ-th input pattern[1] and $\mathbf{d}_\mu \in \mathbb{R}^K$ is the corresponding target class label, with K denoting the number of classes. For the labels, we assume an 1-of-K encoding scheme, i.e. for each label vector

[1] First component of $\mathbf{x}_\mu$ is equal to 1 in order to include the bias.

$\mathbf{d}_\mu$, the component whose index corresponds to the class of pattern $\mathbf{x}_\mu$ is set to "+1", while the other $K-1$ components are set to "-1".

Then, let us randomly select N_1 ($N_1 < N$) data pairs from the available data pool and arrange them along the columns of the matrices $\mathbf{D}$ and $\mathbf{X}$ as follows:

$$\mathbf{X} = [\mathbf{x}_1 \mid \mathbf{x}_2 \mid \cdots \mid \mathbf{x}_{N_1}] \quad \text{and} \quad \mathbf{D} = [\mathbf{d}_1 \mid \mathbf{d}_2 \mid \cdots \mid \mathbf{d}_{N_1}]. \tag{1}$$

where $\dim(\mathbf{X}) = (p+1) \times N_1$ and $\dim(\mathbf{D}) = m \times N_1$. Our goal is to use the matrices $\mathbf{X}$ and $\mathbf{D}$ to build the following linear mapping:

$$\mathbf{D} = \boldsymbol{\beta}\mathbf{X} \quad \text{(batch recall)} \quad \text{or} \quad \mathbf{d}_\mu = \boldsymbol{\beta}\mathbf{x}_\mu \quad \text{(pattern-by-pattern recall)}, \tag{2}$$

for $\mu = 1, \ldots, N_1$. For both recall modes, the dimension of $\boldsymbol{\beta}$ is $K \times (p+1)$.

The ordinary least squares (OLS) solution of the linear system in Eq. (2) is given by the Moore-Penrose generalized inverse as follows:

$$\hat{\boldsymbol{\beta}} = \mathbf{D}\mathbf{X}^T \left(\mathbf{X}\mathbf{X}^T\right)^{-1}, \tag{3}$$

where the hat symbol ($\wedge$) indicates an estimate of the matrix operator $\boldsymbol{\beta}$. A minimum-norm solution for Eq. (2) is given by the regularized version of Eq. (3):

$$\hat{\boldsymbol{\beta}} = \mathbf{D}\mathbf{X}^T \left(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}\right)^{-1}, \tag{4}$$

where $\mathbf{I}$ is the identity matrix of dimension $(p+1) \times (p+1)$ and λ is a very small positive regularization parameter.

Once we have computed $\hat{\boldsymbol{\beta}}$, the remaining $N_2 = N - N_1$ data pairs are used to validate the model. In this regard, for the pattern-by-pattern recall mode, the output of the OLAM is given by

$$\mathbf{y}_\mu = \hat{\boldsymbol{\beta}}\mathbf{x}_\mu, \tag{5}$$

while for the batch recall mode we have $\tilde{\mathbf{Y}} = \hat{\boldsymbol{\beta}}\tilde{\mathbf{X}}$.

The predicted class index i_μ^* for the μ-th testing input pattern is then given by the following decision rule:

$$i_\mu^* = \arg \max_{i=1,\ldots,K} \{y_{i\mu}\} = \arg \max_{i=1,\ldots,K} \{\hat{\boldsymbol{\beta}}_i^T \mathbf{x}_\mu\}, \tag{6}$$

where $y_{i\mu}$ is the i-th component of vector $\mathbf{y}_\mu$ computed as in Eq. (5), with the vector $\boldsymbol{\beta}_i^T$ being the i-th row of the matrix $\hat{\boldsymbol{\beta}}$.

It is worth noting that the parameter vector $\boldsymbol{\beta}_i \in \mathbb{R}^m$, $i = 1, \ldots, m$, can be computed individually by means of the following equation:

$$\hat{\boldsymbol{\beta}}_i = \left(\mathbf{X}\mathbf{X}^T\right)^{-1} \mathbf{X}\mathbf{D}_i^T, \tag{7}$$

where the vector $\mathbf{D}_i$ corresponds to the i-th row of the matrix $\mathbf{D}$.

3 Basics of *M*-Estimation

An important feature of OLS is that it assigns the same importance to all error samples, i.e. all errors contribute the same way to the final solution. A common approach to handle this problem consists in removing outliers from data and then try the usual least-squares fit. A more principled approach, known as *robust regression*, uses estimation methods not as sensitive to outliers as the OLS.

Huber [11] introduced the concept of M-estimation, where M stands for "maximum likelihood" type, where robustness is achieved by minimizing another function than the sum of the squared errors. Based on Huber theory, a general M-estimator applied to the i-th output neuron of the OLAM classifier minimizes the following objective function:

$$J(\boldsymbol{\beta}_i) = \sum_{\mu=1}^{N} \rho(e_{i\mu}) = \sum_{\mu=1}^{N} \rho(d_{i\mu} - y_{i\mu}) = \sum_{\mu=1}^{N} \rho(d_{i\mu} - \boldsymbol{\beta}_i^T \mathbf{x}_\mu), \tag{8}$$

where the function $\rho(\cdot)$ computes the contribution of each error $e_{i\mu} = d_{i\mu} - y_{i\mu}$ to the objective function, $d_{i\mu}$ is the target value of the i-th output neuron for the μ-th input pattern $\mathbf{x}_\mu$, and $\boldsymbol{\beta}_i$ is the weight vector of the i-th output neuron. The OLS is a particular M-estimator, achieved when $\rho(e_{i\mu}) = e_{i\mu}^2$. It is desirable that the function ρ possesses the following properties:

Property 1: $\rho(e_{i\mu}) \geq 0$.
Property 2: $\rho(0) = 0$.
Property 3: $\rho(e_{i\mu}) = \rho(-e_{i\mu})$.
Property 4: $\rho(e_{i\mu}) \geq \rho(e_{i'\mu})$, for $|e_{i\mu}| > |e_{i'\mu}|$.

Parameter estimation is defined by the estimating equation which is a weighted function of the objective function derivative. Let $\psi = \rho'$ to be the derivative of ρ. Differentiating ρ with respect to the estimated weight vector $\hat{\boldsymbol{\beta}}_i$, we have

$$\sum_{\mu=1}^{N} \psi(y_{i\mu} - \hat{\boldsymbol{\beta}}_i^T \mathbf{x}_\mu)\mathbf{x}_\mu^T = \mathbf{0}, \tag{9}$$

where $\mathbf{0}$ is a $(p+1)$-dimensional row vector of zeros. Then, defining the weight function $w(e_{i\mu}) = \psi(e_{i\mu})/e_{i\mu}$, and let $w_{i\mu} = w(e_{i\mu})$, the estimating equations are given by

$$\sum_{\mu=1}^{n} w_{i\mu}(y_{i\mu} - \hat{\boldsymbol{\beta}}_i^T \mathbf{x}_\mu)\mathbf{x}_\mu^T = \mathbf{0}. \tag{10}$$

Thus, solving the estimating equations corresponds to solving a weighted least-squares problem, minimizing $\sum_\mu w_{i\mu}^2 e_{i\mu}^2$.

It is worth noting, however, that the weights depend on the residuals (i.e. estimated errors), the residuals depend upon the estimated coefficients, and the estimated coefficients depend upon the weights. As a consequence, an iterative

estimation method called *iteratively reweighted least-squares* (IRLS) [7] is commonly used. The steps of the IRLS algorithm in the context of training the OLAM classifier using Eq. (7) as reference are described next.

IRLS Algorithm for OLAM Training

Step 1 - Provide an initial estimate $\hat{\boldsymbol{\beta}}_i(0)$ using the regularized least-squares solution in Eq. (7).

Step 2 - At each iteration t, compute the residuals from the previous iteration $e_{i\mu}(t-1)$, $\mu = 1, \ldots, N$, associated with the i-th output neuron, and then compute the corresponding weights $w_{i\mu}(t-1) = w[e_{i\mu}(t-1)]$.

Step 3 - Solve for new weighted-least-squares estimate of $\boldsymbol{\beta}_i(t)$:

$$\hat{\boldsymbol{\beta}}_i(t) = \left[\mathbf{X}\mathbf{W}(t-1)\mathbf{X}^T\right]^{-1}\mathbf{X}\mathbf{W}(t-1)\mathbf{D}_i^T, \tag{11}$$

where $\mathbf{W}(t-1) = \text{diag}\{w_{i\mu}(t-1)\}$ is an $N \times N$ weight matrix. Repeat Steps 2 and 3 until the convergence of the estimated coefficient vector $\hat{\boldsymbol{\beta}}_i(t)$.

Several weighting functions for the M-estimators can be chosen, such as the Huber's weighting function:

$$w(e_{i\mu}) = \begin{cases} \frac{k}{|e_{i\mu}|}, & \text{if } |e_{i\mu}| > k \\ 1, & \text{otherwise.} \end{cases} \tag{12}$$

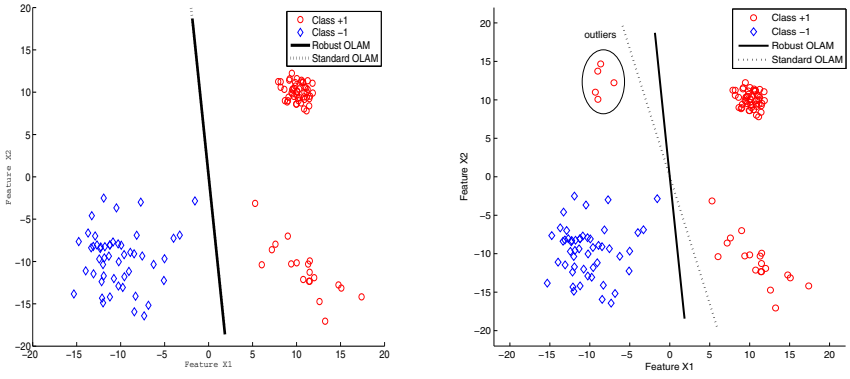
where the parameter k is a tuning constant. Smaller values of k produce more resistance to outliers, but at the expense of lower efficiency when the errors are normally distributed. In particular, $k = 1.345\sigma$ for the Huber function, where σ is a robust estimate of the standard deviation of the errors[2].

In a sum, the basic idea of the proposed approach is very simple: replace the OLS estimation of the weight matrix $\hat{\boldsymbol{\beta}}$ described in Eq. (3) with the one provided by the combined use of the M-estimation framework and the IRLS algorithm. From now on, we refer to the proposed approach by *Robust OLAM* classifier (or ROLAM, for short). In the next section we present and discuss the results achieved by the ROLAM classifier on synthetic and real-world datasets.

4 Simulations and Discussion

As a proof of concept, in the first experiment we aim at showing the influence of outliers in the final position the decision line between two linear separable data classes. For this purpose, we created a synthetic two-dimensional dataset consisting of $N = 120$ samples plus N_{out} outliers. The OLAM and the ROLAM classifiers are trained twice. The first time they are trained with the outlier-free dataset. The second time, they are trained with the outliers added to the original

[2] A usual approach is to take $\sigma = \text{MAR}/0.6745$, where MAR is the median absolute residual.



(a) Dataset without outliers. (b) Dataset with outliers.

Fig. 1. Decision lines of the standard OLAM and the proposed robust OLAM classifiers. (a) Dataset without outliers. (b) Dataset with outliers.

dataset. It is worth mentioning that all data samples are used for training the classifiers, since the goal is to visualize the final position of the decision line and not to compute recognition rates.

For this experiment, the Huber weighting function was used for implementing the ROLAM classifier and the regularization constant required for implementing the standard OLAM classifier was set to $\lambda = 10^{-2}$. The default tuning parameter k of Matlab's `robustfit` function was used. In order to evaluate the final decision lines of the OLAM and the ROLAM classifiers in the presence of outliers, we added $N_{out} = 10$ outliers to the dataset and labelled them as beloging to class $+1$. The outliers were located purposefully far from the class boundary found for the outlier-free case; more specifically, at the decision region of class -1.

The results for the training without outliers are shown in Fig. 1a, where as expected the decision lines of both classifiers coincide. The results for the training with outliers are shown in Fig. 1b, where this time the decision line of the OLAM classifier moved towards the outliers, while the decision line of the ROLAM classifier remained at the same position, thus revealing the robustness of the proposed approach to the presence of outliers. The dataset (with and without outliers) used in the first experiment can be made available by the authors upon request.

In the second and third experiments we aim at evaluating the robustness of the ROLAM classifier using real-world datasets. In these experiments, four weighting functions were tested for implementing the ROLAM classifier and the regularization constant required for implementing the standard OLAM classifier

Table 1. Performance comparison of OLAM and ROLAM classifiers (Iris dataset)

Classifier	$N_{out} = 0\%$	$N_{out} = 5\%$	$N_{out} = 10\%$	$N_{out} = 20\%$	$N_{out} = 30\%$
OLAM	93.05 ± 4.76	91.75 ± 5.52	85.60 ± 9.03	66.60 ± 11.78	56.25 ± 10.65
ROLAM (Bisquare)	93.35 ± 4.55	$\mathbf{94.05 \pm 4.36}$	$\mathbf{93.40 \pm 5.02}$	69.10 ± 11.90	$\mathbf{57.80 \pm 11.20}$
ROLAM (Fair)	93.25 ± 5.52	93.35 ± 5.13	92.05 ± 5.73	$\mathbf{70.20 \pm 11.28}$	53.15 ± 12.30
ROLAM (Huber)	$\mathbf{94.20 \pm 4.75}$	93.10 ± 5.31	93.10 ± 5.49	67.00 ± 10.96	53.35 ± 11.55
ROLAM (Logistic)	93.55 ± 4.68	93.50 ± 5.29	93.30 ± 5.23	68.95 ± 9.33	56.80 ± 11.77

was set to $\lambda = 10^{-2}$. The default tuning parameter k of Matlab's `robustfit` function was adopted for all weighting functions.

In order to evaluate the classifier's robustness to outliers we follow the methodology introduced by Kim and Ghahramani [14]. Thus, the original labels of some data samples of a given class are deliberately changed to the label of the other class. Two datasets were selected, Iris and Vertebral Column (VC), which are publicly available for download from the UCI Machine Learning Repository website [8].

For the Iris dataset, we labelled the samples of classes Virginica ($N_{vir} = 50$ samples) and Versicolor ($N_{ver} = 50$ samples) as $+1$ and -1, respectively. Data from category Setosa ($N_{set} = 50$ samples) will be labelled as belonging to class $+1$, i.e. will be treated as outliers of class $+1$. A certain number N_{out} of outliers are randomly selected and added to the training set of the classifiers. Thus, the total number of training samples is given by $N = P_{train} \times (N_{ver} + N_{vir}) + N_{out}$, where P_{train} is a percentage of samples randomly selected from classes Versicolor and Virginica. We evaluate the performance of the OLAM and ROLAM classifiers for increasing values of N_{out}.

The results are given in Table 1 for different values of N_{out} and for different weighting functions. In this table, we show the values of the classification rates and the corresponding standard deviations averaged over 100 training/testing runs. By analyzing the results, we verify that the ROLAM classifier always performed better than the OLAM, even for the case without outliers ($N_{out} = 0\%$). The ROLAM classifier using the bisquare function achieved the best overall performance. For $N_{out} = 20\%$, the performances of the ROLAM classifier using the bisquare and the fair functions are statistically equivalent. It is worth noting that the standard deviation of the classification rate increases with the increase in N_{out}. Also interesting is the fact that, for $N_{out} > 30\%$, the ROLAM classifier performs as badly as the OLAM classifier (results are not shown here for lack of space). However, when the number of outliers is too high, perhaps they should not be considered as outliers anymore, but as usual data samples of the class. In this situation, we recommend a more powerful classifier (e.g. the ELM) to be used, since it can produce a curved decision surface.

For the VC dataset, we labelled the samples of classes Normal ($N_{nor} = 100$ samples) and Spondylolisthesis[3] ($N_{spl} = 150$ samples) as $+1$ and -1, respectively. Samples from category Spondylolisthesis will be labelled as belonging to

[3] Spondylolisthesis is the displacement of a vertebra or the vertebral column in relation to the vertebrae below.

Table 2. Performance comparison of OLAM and ROLAM classifiers (VC dataset)

Classifier	$N_{out} = 0\%$	$N_{out} = 5\%$	$N_{out} = 10\%$	$N_{out} = 20\%$	$N_{out} = 30\%$
OLAM	90.70 ± 4.05	85.34 ± 5.18	79.80 ± 6.12	63.66 ± 7.79	53.66 ± 7.30
ROLAM (Bisquare)	$\mathbf{91.98 \pm 3.54}$	$\mathbf{94.32 \pm 2.95}$	$\mathbf{87.92 \pm 5.21}$	$\mathbf{67.00 \pm 6.11}$	$\mathbf{54.74 \pm 7.79}$
ROLAM (Fair)	91.54 ± 3.58	89.16 ± 4.43	82.36 ± 5.47	63.52 ± 7.16	52.76 ± 7.47
ROLAM (Huber)	91.58 ± 3.61	90.14 ± 4.29	84.26 ± 5.88	64.42 ± 6.77	53.88 ± 8.82
ROLAM (Logistic)	91.76 ± 3.58	90.10 ± 4.25	84.62 ± 5.35	64.14 ± 7.53	51.84 ± 8.83

class +1, i.e. will be treated as outliers of class +1. For generating the training set, we first compute the centroids of both classes. Then, we select randomly a certain number of samples from the total available (e.g. $0.8 \times (N_{norm} + N_{spl})$). Finally, among the selected samples of class Spondylolisthesis, we select a certain quantity (N_{out}) of the most distant ones to the centroid of the class Normal to have their labels changed to +1.

The results are given in Table 2 for different values of N_{out} and for different weighting functions. In this table, we show the values of the classification rates and the corresponding standard deviations averaged over 100 training/testing runs. One can easily note that the ROLAM classifier using the bisquare function performed much better than the standard OLAM classifier, even for the case without outliers. Again, when the percentage of outliers reaches high values (e.g. $N_{out} \geq 20\%$) the performances of both classifiers begin to deteriorate considerably, with the performance of the ROLAM classifier degrading at a lower rate.

In the previous experiments we used the default value of the tuning parameter k. Since this is a free parameter, we show in a final experiment that the performance of the ROLAM classifier can be improved considerably if an optimal tuning parameter is searched during the training phase. In this experiment we used the Wisconsin Breast Cancer (Diagnostic) dataset, which is also publicly available for download from the UCI Machine Learning Repository website [8]. The range of the search for the optimal value of the tuning parameter for each weighting function covered the interval from 0.1 to 10.

The results are shown in Table 3, where k_{def} and k_{opt} denote the default and the optimal values of the tuning parameter, respectively. In this table, we show the values of the classification rates and the corresponding standard deviations averaged over 100 training/testing runs. Below each value of the pair (classification rate, standard deviation) we show the associated value of k_{def} or k_{opt} for a specific percentage of outliers. We labelled the samples of class Malignant as -1 and of class Benign as -1. During training a certain number of randomly selected samples from the category Benign (class -1) will be labelled as belonging to class malignant (class +1), i.e. will be treated as outliers of class -1. For generating the outliers we followed the same procedure used in the second and third experiments. The regularization constant required for implementing the standard OLAM classifier was again set to $\lambda = 10^{-2}$.

The results in Table 3 emphasize the power of the M-estimation method in providing a principled approach for robust pattern classification. It is easy to

Table 3. Performance comparison of OLAM and ROLAM classifiers (Breast Cancer dataset)

Classifier	0%	5%	10%	20%	30%
OLAM	95.39±1.87	92.47±2.51	85.76±3.31	74.58±4.45	62.12±4.50
ROLAM (**Andrews**)	94.33±1.96	95.24±1.83	86.50±3.27	75.02±3.86	62.94±4.48
	($k_{def} = 1.339$)	($k_{def} = 1.339$)	($k_{def} = 1.339$)	($k_{def} = 1.339$)	($k_{def} = 1.339$)
	95.89±1.80	95.40±2.00	89.75±5.09	80.46±4.04	66.85±4.60
	($k_{opt} = 3.5$)	($k_{opt} = 1.5$)	($k_{opt} = 1$)	($k_{opt} = 0.5$)	($k_{opt} = 0.5$)
ROLAM (**Bisquare**)					
	94.78±1.96	95.17±1.88	85.69±3.38	75.29±4.15	62.76±4.22
	($k_{def} = 4.685$)	($k_{def} = 4.685$)	($k_{def} = 4.685$)	($k_{def} = 4.685$)	($k_{def} = 4.685$)
	95.57±1.77	95.31±1.85	93.61±2.78	81.44±4.79	70.37±5.30
	($k_{opt} = 9.5$)	($k_{opt} = 4.5$)	($k_{opt} = 3$)	($k_{opt} = 1.5$)	($k_{opt} = 1$)
ROLAM (**Cauchy**)					
	94.89±1.69	93.59±2.66	86.12±3.51	75.47±3.93	62.44±5.00
	($k_{def} = 2.385$)	($k_{def} = 2.385$)	($k_{def} = 2.385$)	($k_{def} = 2.385$)	($k_{def} = 2.385$)
	95.82±1.91	94.09±2.23	90.72±2.72	77.53±4.14	69.82±5.02
	($k_{opt} = 8.5$)	($k_{opt} = 2$)	($k_{opt} = 0.5$)	($k_{opt} = 0.1$)	($k_{opt} = 0.1$)
ROLAM (Fair)					
	94.80±1.94	92.55±2.43	85.88±3.19	75.96±3.80	62.60±5.06
	($k_{def} = 1.400$)	($k_{def} = 1.400$)	($k_{def} = 1.400$)	($k_{def} = 1.400$)	($k_{def} = 1.400$)
	95.61±1.69	93.40±2.11	87.17±2.65	76.03±3.89	64.94±4.46
	($k_{opt} = 9$)	($k_{opt} = 0.1$)	($k_{opt} = 0.1$)	($k_{opt} = 4.5$)	($k_{opt} = 0.1$)
ROLAM (Huber)					
	94.07±2.00	93.94±2.27	86.21±3.07	74.61±3.57	62.82±4.68
	($k_{def} = 1.345$)	($k_{def} = 1.345$)	($k_{def} = 1.345$)	($k_{def} = 1.345$)	($k_{def} = 1.345$)
	95.87±1.78	93.71±2.33	87.91±3.04	76.61±3.61	65.65±5.07
	($k_{opt} = 7.5$)	($k_{opt} = 1.5$)	($k_{opt} = 0.1$)	($k_{opt} = 0.5$)	($k_{opt} = 0.1$)
ROLAM (Logistic)					
	94.68±2.12	92.97±2.43	86.23±3.14	75.79±4.10	64.29±5.16
	($k_{def} = 1.205$)	($k_{def} = 1.205$)	($k_{def} = 1.205$)	($k_{def} = 1.205$)	($k_{def} = 1.205$)
	95.86±1.60	93.29±2.55	86.46±3.39	76.16±3.90	65.20±6.10
	($k_{opt} = 6.5$)	($k_{opt} = 1$)	($k_{opt} = 1.5$)	($k_{opt} = 2$)	($k_{opt} = 0.1$)
ROLAM (**Talwar**)					
	95.54±1.86	95.02±2.15	85.82±3.31	74.76±3.77	62.16±5.33
	($k_{def} = 2.795$)	($k_{def} = 2.795$)	($k_{def} = 2.795$)	($k_{def} = 2.795$)	($k_{def} = 2.795$)
	95.99±1.80	95.44±1.90	92.19±2.79	78.75±3.69	68.01±8.55
	($k_{opt} = 5$)	($k_{opt} = 2.5$)	($k_{opt} = 1.5$)	($k_{opt} = 1$)	($k_{opt} = 0.5$)
ROLAM (**Welsch**)					
	94.61±1.93	94.89±1.98	85.63±3.25	74.97±3.87	62.19±5.46
	($k_{def} = 2.985$)	($k_{def} = 2.985$)	($k_{def} = 2.985$)	($k_{def} = 2.985$)	($k_{def} = 2.985$)
	95.70±1.72	94.94±1.89	91.81±3.05	80.45±4.62	69.73±4.91
	($k_{opt} = 8$)	($k_{opt} = 2.5$)	($k_{opt} = 1.5$)	($k_{opt} = 1$)	($k_{opt} = 0.5$)

note that the performances of the ROLAM classifier are much better than those achieved by the OLAM classifier and by the ROLAM classifier using default values of the tuning parameter, specially in the presence of a high number of outliers (10%, 20% and 30%). The improvement is particularly sharp for the following weighting functions: Andrews, Bisquare, Cauchy, Talwar and Welsch.

5 Conclusion

In this paper we introduced a robust OLAM classifier for supervised pattern classification in the presence of labeling errors (outliers) in the data set. The robust OLAM classifier is designed by means of M-estimation methods which are used to compute the weight matrix operator instead of using the ordinary least

squares solution. We have shown that the resulting classifier is robust to label noise not only near the class boundaries, but also far from the class boundaries which can result from mistakes in labelling or gross errors in measuring the input features.

Currently we are evaluating the use of M-estimation techniques in the design of robust ELM-based classifiers. The results we obtained so far suggests that this is a promising approach, since the ELM is in fact using the OLAM classifier in the output layer.

References

1. Baek, D., Oh, S.Y.: Improving optimal linear associative memory using data partitioning. In: Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2006), vol. 3, pp. 2251–2256 (2006)
2. Barreto, G.A., Frota, R.A.: A unifying methodology for the evaluation of neural network models on novelty detection tasks. Pattern Analysis and Applications 16(1), 83–972 (2013)
3. Cherkassky, V., Fassett, K., Vassilas, N.: Linear algebra approach to neural associative memories and noise performance of neural classifiers. IEEE Transactions on Computers 40(12), 1429–1435 (1991)
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley & Sons (2006)
5. Eichmann, G., Kasparis, T.: Pattern classification using a linear associative memory. Pattern Recognition 22(6), 733–740 (1989)
6. Emmerich, C., Reinhart, R.F., Steil, J.J.: Recurrence enhances the spatial encoding of static inputs in reservoir networks. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) ICANN 2010, Part II. LNCS, vol. 6353, pp. 148–153. Springer, Heidelberg (2010)
7. Fox, J.: Applied Regression Analysis, Linear Models, and Related Methods. Sage Publications (1997)
8. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
9. Horata, P., Chiewchanwattana, S., Sunat, K.: Robust extreme learning machine. Neurocomputing 102, 31–44 (2012)
10. Huang, G.B., Wang, D.H., Lan, Y.: Extreme learning machines: a survey. International Journal of Machine Learning and Cybernetics 2, 107–122 (2011)
11. Huber, P.J.: Robust estimation of a location parameter. Annals of Mathematical Statistics 35(1), 73–101 (1964)
12. Huber, P.J., Ronchetti, E.M.: Robust Statistics. John Wiley & Sons, LTD. (2009)
13. Hunt, B., Nadar, M., Keller, P., VonColln, E., Goyal, A.: Synthesis of a nonrecurrent associative memory model based on a nonlinear transformation in the spectral domain. IEEE Transactions on Neural Networks 4(5), 873–878 (1993)
14. Kim, H.-C., Ghahramani, Z.: Outlier robust gaussian process classification. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSPR&SPR 2008. LNCS, vol. 5342, pp. 896–905. Springer, Heidelberg (2008)
15. Kohonen, T., Oja, E.: Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. Biological Cybernetics 25, 85–95 (1976)

16. Kohonen, T., Ruohonen, M.: Representation of associated data by matrix operators. IEEE Transactions on Computers 22(7), 701–702 (1973)
17. Li, D., Han, M., Wang, J.: Chaotic time series prediction based on a novel robust echo state network. IEEE Transactions on Neural Networks and Learning Systems 23(5), 787–799 (2012)
18. Poggio, T., Girosi, F.: Networks for approximation and learning. Proceedings of the IEEE 78(9), 1481–1497 (1990)
19. Stiles, G.S., Denq, D.: On the effect of noise on the Moore-Penrose generalized inverse associative memory. IEEE Transactions on Pattern Analysis and Machine Intelligence 7(3), 358–360 (1985)
20. Stiles, G., Denq, D.L.: A quantitative comparison of the performance of three discrete distributed associative memory models. IEEE Transactions on Computers 36(3), 257–263 (1987)
21. Webb, A.: Statistical Pattern Recognition, 2nd edn. John Wiley & Sons, LTD. (2002)